TP 13436E

ICE NAVIGATION SIMULATION – MAIN SIMULATOR PLATFORM (MSP)

Prepared for
Transportation Development Centre
Transport Canada

by
PhiloSoft Inc.
3608 Boulevard St. Charles, Suite #27c
Kirkland, Quebec
H9H 3C3

May 1999

103-04684/

TP 13436E

ICE NAVIGATION SIMULATION – MAIN SIMULATOR PLATFORM (MSP)

by
Charles G. Marton
PhiloSoft Inc.

May 1999

This report reflects the views of PhiloSoft Inc. and not necessarily those of the Transportation Development Centre of Transport Canada or the sponsoring organizations.

The Transportation Development Centre does not endorse products or manufacturers. Trade or manufacturers' names appear in this report only because they are essential to its objectives.

Un sommaire français se trouve avant la table des matières.

Transport Canada / Transports Canada

PUBLICATION DATA FORM

1. Transport Canada Publication No.	2. Project No.	3. Recipient's Catalogue No.
TP 13436E	9120	

4. Title and Subtitle	5. Publication Date
Ice Navigation Simulation – Main Simulator Platform (MSP)	May 1999
	6. Performing Organization Document No.

7. Author(s)	8. Transport Canada File No.
Charles G. Marton	ZCD1460-368

9. Performing Organization Name and Address	10. PWGSC File No.
PhiloSoft Inc. 3608 St. Charles Blvd., #27c Kirkland, Quebec H9H 3C3	XSD-6-01997
	11. PWGSC or Transport Canada Contract No.
	T8200-6-6549

12. Sponsoring Agency Name and Address	13. Type of Publication and Period Covered
Transportation Development Centre (TDC) 800 René Lévesque Blvd. West Suite 600 Montreal, Quebec H3B 1X9	Final
	14. Project Officer
	Charles Gautier

15. Supplementary Notes (Funding programs, titles of related publications, etc.)

Co-sponsored by Prairie and Northern Region and the Program of Energy Research and Development (PERD)

16. Abstract

Navigating ships in ice-infested waters has always been a risky and dangerous task. The development of an ice navigation simulator stems from a study conducted on behalf of the Canadian Coast Guard (TP 12496), in which the requirements for a training course were identified.

Phase 1 demonstrated the feasibility of using synthetic aperture radar (SAR) as a basis for developing an ice navigation simulator. The main simulator platform was developed in conjunction with other modules such as the training aids module and the shipboard radar simulation module.

The main simulation software provides several key elements:

- SAR data conversion tools
- SAR data analysis tools
- Virtual out-of-window bridge view
- A database management system
- Global Positioning System (GPS) and Gyro simulation
- Instructor and student functionality
- Bulletin Board Service data downlink
- Scaleable network operation

All of the modules that make up the simulator were successfully integrated.

17. Key Words	18. Distribution Statement
Marine, radar, simulation, ice navigation	Limited number of copies available from the Transportation Development Centre

19. Security Classification (of this publication)	20. Security Classification (of this page)	21. Declassification (date)	22. No. of Pages	23. Price
Unclassified	Unclassified	—	xiv, 50, app.	Shipping/ Handling

Transports Canada Transport Canada

FORMULE DE DONNÉES POUR PUBLICATION

1. N° de la publication de Transports Canada	2. N° de l'étude	3. N° de catalogue du destinataire
TP 13436E	9120	

4. Titre et sous-titre	5. Date de la publication
Ice Navigation Simulation – Main Simulator Platform (MSP)	Mai 1999
	6. N° de document de l'organisme exécutant

7. Auteur(s)	8. N° de dossier - Transports Canada
Charles G. Marton	ZCD1460-368

9. Nom et adresse de l'organisme exécutant	10. N° de dossier - TPSGC
PhiloSoft Inc. 3608 boul. Saint-Charles, #27c Kirkland, Quebec H9H 3C3	XSD-6-01997
	11. N° de contrat - TPSGC ou Transports Canada
	T8200-6-6549

12. Nom et adresse de l'organisme parrain	13. Genre de publication et période visée
Centre de développement des transports (CDT) 800, boul. René-Lévesque Ouest Bureau 600 Montréal (Québec) H3B 1X9	Final
	14. Agent de projet
	Charles Gautier

15. Remarques additionnelles (programmes de financement, titres de publications connexes, etc.)

Projet coparrainé par la région des Prairies et du Nord ainsi que par le Programme de recherche et développement énergétiques (PRDE)

16. Résumé

La navigation dans les eaux envahies par les glaces a toujours été risquée et dangereuse. Le développement d'un simulateur de navigation dans les glaces découle d'une étude effectuée pour la Garde côtière canadienne (TP 12496) dans laquelle furent établies les exigences d'un cours de formation.

La phase 1 a démontré la faisabilité d'utiliser un radar à ouverture synthétique (ROS) comme base de développement d'un simulateur de navigation dans les glaces. La plate-forme de simulation principale (PSP) a été mise au point conjointement avec d'autres modules, tels que le module des aides à la formation et le module de simulation de radar de bord.

Le logiciel de simulation principal a plusieurs composantes clés :

- outils de conversion de données ROS
- outils d'analyse de données ROS
- vue virtuelle depuis la fenêtre de la passerelle
- système de gestion d'une base de données
- simulation du système de positionnement global (GPS) et d'un gyrolaser
- fonctionnalité instructeur-élève
- liaison descendante pour babillard électronique
- exploitation en réseau évolutive

Tous les modules constitutifs du simulateur ont été intégrés avec succès.

17. Mots clés	18. Diffusion
Marine, radar, simulation, navigation dans les glaces	Le Centre de développement des transports dispose d'un nombre limité d'exemplaires.

19. Classification de sécurité (de cette publication)	20. Classification de sécurité (de cette page)	21. Déclassification (date)	22. Nombre de pages	23. Prix
Non classifiée	Non classifiée	—	xiv, 50, ann.	Port et manutention

ACKNOWLEDGEMENTS

A number of individuals contributed to the completion of this phase of the project. I would like to extend *extra special* thanks to Charles Gautier of the Transportation Development Centre for his many valuable suggestions and thorough review of this and related documentation.

EXECUTIVE SUMMARY

This project was a result of a study conducted on behalf of the Canadian Coast Guard (TP 12496) that outlined the essential elements required to make up an International Ice Navigator Course. A significant component of this course involved training ice navigators with the aid of a simulator. The need for a simulator has been recognized by such countries as Finland, Russia, Sweden, Germany and Norway.

Different approaches have been considered for the implementation of this capability, hereafter called the Ice Navigation Simulator, and a number of facilities exist where large modifiable simulation engines could be used to perform this task. However, these systems use proprietary technologies and are not considered "open" systems. Besides being very expensive, they do not allow for widespread distribution and availability. Given state-of-the-art PC technology, complete with near workstation performance at a fraction of the cost, together with the explosion of multimedia and available virtual reality equipment, it is now possible to consider alternatives to the fixed, high-cost and proprietary systems currently in use for shipboard simulation. Therefore, a new approach to a realistic, low-cost implementation of the simulator was adopted.

The simulator encompasses the following elements:

- Simulation of ice on shipboard radar
- Simulation and management of remotely sensed data
- Simulation of visual aspect of ice, in daylight and night transit conditions
- Ship transit simulation (basic at this stage)
- Ice recognition and ice climatology training aids
- Ice regime entry rules training aid
- Ice Navigation systems and Electronic Charting and Display Information System (ECDIS) support

The Main Simulator Platform (MSP) ties these elements together to create an environment that visually and operationally resembles the ice navigation environment. The simulator was developed to implement a full and complete training platform for ice navigation. The overall program objective was to develop a low-cost, PC-based Ice Navigation Simulation platform to train entry-level ice navigators. The MSP included the development of the:

- scenario generation module (SGM)
- base data management module (BDM) (for SAR and remotely sensed imagery)
- visual simulation module (VSM)
- user interface to the system

The MSP project included the development of interfaces to auxiliary simulation modules such as the shipboard radar simulation module (SRSM) and the transit simulation module (TSM), as well as interfaces to Ice Navigation and ECDIS systems.

In addition, interfaces to other training aid modules (TAMs) were implemented during the course of this project; they included the Ice Recognition and the Ice Regime Entry Rules Training Aids.

Initially, the SAR images were analysed off-line and their features identified and entered into an associate feature file with the aid of the SGM application. These selected SAR data files and their corresponding feature files formed the basis of the simulation. The feature files contained such information as:

- ice type
- ice thickness
- feature boundaries
- specific features such as ice flows, icebergs, land, open water

The SGM uses the feature file and the information in the corresponding SAR image to create a height map for every pixel in the SAR file. This is used for the SRSM. The SGM also exports an ASCII, which the visual generation module (VGM) uses to generate a 3-D model of the world. The visual simulation module (VSM) then takes this 3-D model and displays it on the screen.

In its present form, the MSP, together with the other integrated components, provides the following levels of information to the entry-level trainee:

- Out of Window Bridge Views
- Marine Radar Views
- Ice Navigation Tools Views
- Navigation Instrumentation (GPS/Gyro Compass/Throttle/Rudder)

This phase of the project was a success despite some initial technical difficulties with system performance and speed. All of the subsystems above have been developed to the operational level; however, there is still much work to do. This work includes but is not limited to, the following:

- Improving the automation and level of analysis
- Improving the visual system
- Adding the new transit simulation module
- Adding the ship's track

SOMMAIRE

Ce projet est le résultat d'une étude effectuée pour la Garde côtière canadienne (TP 12496) qui a décrit les éléments essentiels d'un cours international de navigation dans les glaces. La formation des navigateurs au moyen d'un simulateur est une composante importante de ce cours. Le besoin d'un simulateur a été reconnu par des pays tels que la Finlande, la Russie, la Suède, l'Allemagne et la Norvège.

Différentes approches ont été examinées pour la réalisation de ce simulateur de navigation dans les glaces. Il existe au Canada un certain nombre d'installations où des moteurs de simulation modifiables de grande capacité pourraient être utilisés à cette fin. Toutefois, ces systèmes utilisent des technologies brevetées et ne sont pas considérés comme des systèmes «ouverts». En plus d'être très onéreux, ils ne peuvent être distribués facilement ni ne sont facilement disponibles. Avec les ordinateurs perfectionnés actuels, qui offrent une grande performance à un faible coût, vu l'explosion multimédia et la disponibilité d'équipements à réalité virtuelle, il est maintenant possible d'envisager des solutions de rechange aux onéreux systèmes brevetés utilisés actuellement comme simulateurs de navigation. Une nouvelle approche a donc été envisagée pour réaliser un simulateur peu onéreux.

Ce simulateur comporte les fonctions suivantes :

- simulation des glaces sur le radar de bord
- simulation et gestion des données de télédétection
- simulation de l'aspect des glaces durant le jour et durant la nuit
- simulation de passages de navires (élémentaire à l'heure actuelle)
- aide d'apprentissage en matière de reconnaissance des glaces et de climatologie des glaces
- aide d'apprentissage des conditions autorisant l'entrée dans des régimes de glaces
- soutien aux systèmes de navigation dans les glaces et au Système électronique de visualisation des cartes marines (SEVCM).

La plate-forme de simulation principale (PSP) fait le lien entre ces éléments pour créer un environnement qui ressemble visuellement et opérationnellement à l'environnement de navigation dans les glaces. Elle a été élaborée comme plate-forme de formation complète pour la navigation dans les glaces. L'objectif global du programme était de mettre au point une plate-forme de simulation de navigation dans les glaces sur ordinateur personnel peu onéreuse pour former les navigateurs débutants. La création de la PSP a nécessité le développement :

- d'un module de génération de scénarios (MGS)
- d'un module de gestion de données de base (MGDB) (pour les radars à ouverture synthétique et l'imagerie de télédétection)
- d'un module de simulation visuelle (MSV)
- d'une interface utilisateur pour le système.

Le projet portant sur la plate-forme de simulation principale incluait le développement d'interfaces pour les modules de simulation auxiliaires tels que le module de simulation de radar de bord (MSRB) et le module de simulation de passages (MST), ainsi que d'interfaces pour le système de navigation dans les glaces et le SEVCM.

Des interfaces pour d'autres modules d'aide à la formation (MAF) ont également été créées au cours de ce projet, les modules d'aide à la formation sur la reconnaissance des glaces et sur les règles de pénétration dans les glaces entre autres.

Initialement, les images ROS étaient analysées hors ligne et leurs caractéristiques étaient identifiées, puis introduites dans un fichier de caractéristiques connexe à l'aide du MGS. Ces fichiers de données ROS sélectionnées et leurs fichiers de caractéristiques correspondants servaient de base à la simulation. Les fichiers de caractéristiques contenaient des informations telles que :

- le type de la glace
- l'épaisseur de la glace
- les frontières des caractéristiques
- des caractéristiques particulières telles que la circulation des glaces, les icebergs, les terres, les eaux libres.

Le MGS utilise le fichier des caractéristiques et l'information contenue dans l'image ROS correspondante pour créer une table de bits de hauteur pour chaque pixel du fichier ROS. Celle-ci est utilisée par le MSRB. Le MGS exporte également un fichier ASCII que le module de génération visuelle (MGV) utilise pour produire un module tridimensionnel de l'environnement. Le module de simulation visuelle (MSV) affiche ensuite ce modèle tridimensionnel sur l'écran.

Sous sa forme actuelle, la PSP et ses composantes annexes fournissent les niveaux d'information suivants à l'apprenti débutant :

- des vues depuis la fenêtre de la passerelle
- des vues de radar de marine
- des vues d'outils de navigation dans les glaces
- des instruments de navigation (GPS/gyrocompas/commande des gaz/gouvernail)

Cette phase du projet a été un succès en dépit de certaines difficultés techniques initiales concernant la performance et la vitesse du système. Tous les sous-systèmes mentionnés ci-dessus ont atteint le niveau opérationnel. Il reste toutefois beaucoup à faire, entre autres :

- améliorer l'automatisation et le niveau des analyses
- améliorer le système visuel
- ajouter le nouveau module de simulation de passages de navires
- ajouter la trajectoire du navire.

TABLE OF CONTENTS

1. INTRODUCTION 1

1.1. SCOPE 1

1.2. OVERVIEW 1

1.3. DOCUMENT OVERVIEW 1

2. MSP GENERAL ARCHITECTURE 2

3. SYSTEM DESCRIPTION (HARDWARE) 3

4. SYSTEM DESCRIPTION (SOFTWARE) 5

4.1. OFF-LINE DATA PREPARATION 5

4.1.1. SARTOBMP CONVERTER 5

4.1.2. SARTOBMP SOURCE CODE FILES 9

4.1.3. SARTOBMP DATA FLOW AND CONTROL 10

4.1.4. SCENARIO GENERATION MODULE 11

4.1.5. SGM SOURCE CODE FILES 19

4.1.6. SGM DATA FLOW AND CONTROL 20

4.1.7. VISUALS GENERATION MODULE (VGM) 21

4.2. MSP/VSM START-UP 23

4.2.1. MSP 23

4.2.2. DATA FLOW AND CONTROL 25

4.2.3. MSP USER INTERFACE 26

4.2.4. SHIP CONTROL DIALOGS (INSTRUCTOR AND STUDENT) 26

4.2.5. WEATHER CONTROL DIALOG. 27

4.2.6. MSP SOURCE CODE FILES 28

4.2.7. VSM 29

4.2.8. DATA FLOW AND CONTROL 30

4.2.9. VSM USER INTERFACE 31

4.2.10. VSM SOURCE CODE FILES 32

4.2.11. SRSM AND SRSM SERVER 33

4.2.12. MSP – GPS GYRO 33

4.2.13. TRANSIT SIMULATOR MODULE 33

4.2.14. NAVIGATIONAL AIDS–ICENAV AND ECDIS 33

4.3. MSP/VSM RUN TIME 34

4.3.1. VSM RUN TIME SCENARIO 34

4.3.2. MSP – SERVER 34

4.3.3. MSP – BASE DATA MANAGEMENT 35

4.3.4. GPS AND GYRO MESSAGE GENERATION 35

4.3.5. MSP – DOWNLINK 36

5. MSPSERVER 37

5.1. MSP INTER-PROCESS COMMUNICATIONS 37

5.2. MSPSERVER FUNCTIONS 38

5.3. MSPSERVER AND MSPCLIENT.DLL 41
5.4. MSPSERVER IMPLEMENTATION 42
6. TYPICAL OPERATIONAL USE 43
6.1. VIA SATELLITE AND REMOTELY SENSED DATA 45
6.2. VIA MARINE RADAR 46
6.3. INCLOSE (OUT OF WINDOW) 48
7. CONCLUSION AND RECOMMENDATIONS 49
APPENDIX A – ADDITIONAL MSP/VSM SAMPLES

LIST OF TABLES

Table 4.1.2-1 SARtoBMP Source File Name List 9
Table 4.1.5-1 SGM Source File Name List 19
Table 4.1.7-1 VGM Software Components 21
Table 4.2.6-1 MSP Source File Name List 28
Table 4.2.10-1 VSM Source File Name List 32

LIST OF FIGURES

Figure 2-1 Simulator Applications and DLLs 2
Figure 3-1 PC Architecture 4
Figure 4.1.3-1 SARtoBMP Data and Control Diagram 10
Figure 4.1.6-1 SGM Data and Control Diagram 20
Figure 4.2.1-1 MSP Start-Up Dialogue 24
Figure 4.2.2-1 MSP Control and Data Flow 25
Figure 4.2.3-1 Main Simulator Platform GUI 26
Figure 4.2.4-1 Ship Control Dialog 27
Figure 4.2.5-1 Weather Control Dialog 27
Figure 4.2.8-1 VSM Data Flow and Control Diagram 30
Figure 4.2.9-1 VSM User Interface 31
Figure 4.3.1-1 VSM Run Time Scenario Diagram 34
Figure 5.1-1 MSP Server and Client(s) Communication Architecture 37
Figure 5.2-1 MSP Function Calls Between MSP Server and MSP Clients 38
Figure 5.3-1 MSP Function Calls Between MSP Server and MSP Clients 41
Figure 5.4-1 MSP Client Server Implementation 42
Figure 6-1 MSP Typical Use Example 44
Figure 6.1-1 IceNav32 Route Planning Example 45
Figure 6.2-1 SRSM Image #1 46
Figure 6.2-2 SRSM Image #2 47
Figure 6.3-1 Inclose "out of window view" 48

1. Introduction

1.1. Scope

This document contains the final report for the Ice Navigation – Main Simulator Platform (MSP) Project.

1.2. Overview

The Main Simulator Platform (MSP) was developed in order that a full and complete training platform could be implemented for ice navigation. The overall program objective was to develop a low-cost PC-based Ice Navigation Simulation platform to train entry-level ice navigators. The MSP included the development of a scenario generation module (SGM); the base data management module (BDM), for Synthetic Aperture Radar (SAR) and remotely sensed imagery; the visual simulator module (VSM); and the user interface to the system. The MSP project included the development of interfaces to auxiliary simulator modules such as the shipboard radar simulation module (SRSM), the transit simulator module (TSM) as well as interfaces to the Ice Navigation system and the electronic charting and display information system (ECDIS). In addition, the interfaces to various other training aid modules (TAMs) were implemented during the course of this project. These included the Ice Recognition and the Ice Regime Entry Rules Training Aids.

1.3. Document Overview

This document contains a brief description of the general architecture of the MSP as well as its functionality, the general data and the data exchange mechanisms. In addition, it addresses the achieved system performance and provides a user interface and mockup. Note: This document deals with specifying the detailed functionality of the components within the MSP and not that of the subsystems furnished by other suppliers.

The MSP operation can be divided into three principle areas. They are:

1. The off-line preparation of the simulator data, which includes the SAR data conversion, the scenario, and visual world generation,

2. The system initialization, which includes identifying the initial ship name, operator, weather conditions, time of day, etc.,

3. The system run-time operation, which includes the dynamic behavior of the simulator as the ship moves through the virtual ice world.

This document also provides several "views" of the MSP. They include:

- a context diagram and descriptions,
- functional process diagrams,
- data and control flow diagrams,
- source code lists. (The actual source code was delivered under separate cover.)

2. MSP General Architecture

The MSP provides the data to and interfaces with various subsystems. The MSP software is composed of several components and dynamic link libraries. Figure 2-1 illustrates the relationship of the various software components that make up the Ice Navigation Simulator. The components are described in detail in the sections that follow.

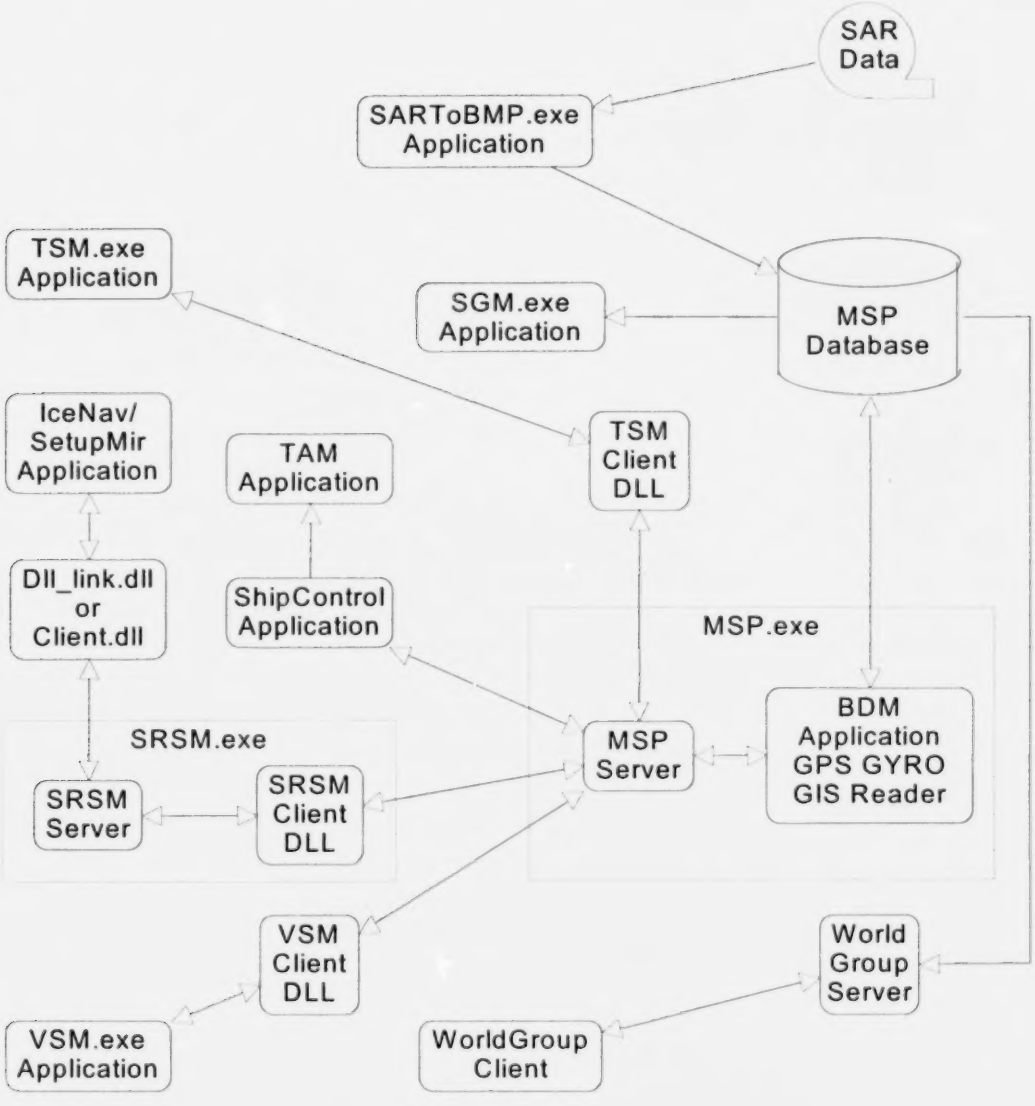


Figure 2-1 Simulator Applications and DLLs

3. System Description (Hardware)

The MSP is a distributed system whose software runs on several networked platforms. This architecture was chosen because, as PC hardware becomes faster and more economical, the entire application will eventually be able to run on a single PC.

For now, a minimum of two PCs are required to run the MSP. A third PC is required to run client applications. The PC allocation is as follows.

- one PC to run the Ice Navigation System and/or ECDIS system,
- one PC for the MSP and Radar Servers,
- one PC for the Visual System.

The computers are connected via a 10-base T network hub (See Figure 3-1). A 100-base T network hub could be used instead but is not necessary for the required performance. The star network topology was selected since it is inherently faster than clothesline architecture whose likelihood of network collision is much higher.

Originally, three Matrox Millennium Graphics cards were purchased for this MSP. After some software prototyping, it was determined that these cards were just too slow for the level of performance required. Subsequent investigation revealed that an OPENGL library could be used together with the Primary Image Graphics Cards. This approach worked but required the purchase of the Piranha P10 (~$8K USD), which was not allocated or budgeted in the proposal. The reason the boards are so expensive is that the OPENGL is done on board. At present the Tempest Library supports up to eight video channels with the Piranha P-10 boards.

As the project progressed we elected to use the Obsidian 100 (~$350 USD) with good graphics but it required a more powerful host. This approach offered a good balance between performance and cost. The only drawback at this stage is the VSM/VGM applications will need to be rewritten in OPENGVS in order to support the operation of more channels from the cheaper graphics boards.

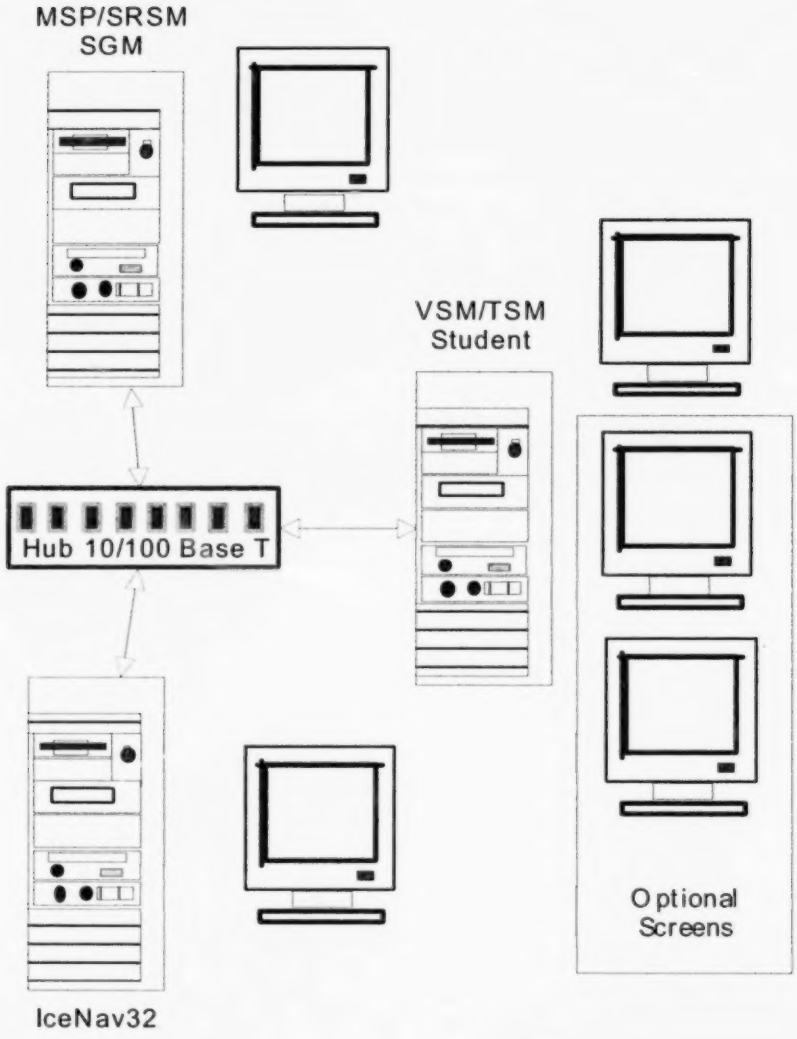


Figure 3-1 PC Architecture

4. System Description (Software)

The following section describes the various levels of operation of the MSP software.

4.1. Off-line Data Preparation

The simulator is based on SAR and feature data that was analyzed manually. The SAR data has the correct spatial relationship between flows but has no ice classification data or height information. This information was added with the aid of ice specialists from Enfotec and utilities developed by PhiloSoft. AES Ice Center in Ottawa provided the original SAR data files. There were over 700 available flight lines to choose from. From these, two were selected by Ice Centre as good examples. One was from the Gulf in February. The other was Parry Sound in May. The data resolution was 25 m per pixel for both data files.

Because of the volume of data used it was necessary to prepare the MSP data ahead of time. Initial experimentation with trying to dynamically create the various height bitmaps proved impossible given the available CPU processing power. This was particularly true for the radar simulator where bitmaps for the individual radar setting were required for both the "zoomed" SAR and the corresponding height maps. Currently it takes approximately two days to generate all the height bitmaps for the system. Another point to mention is the level of analysis. It was determined that the manual analysis of ~ 750 features took ~ five working days. This will need to be researched further in future to see if better mechanisms can be put into place.

4.1.1. SARtoBMP Converter

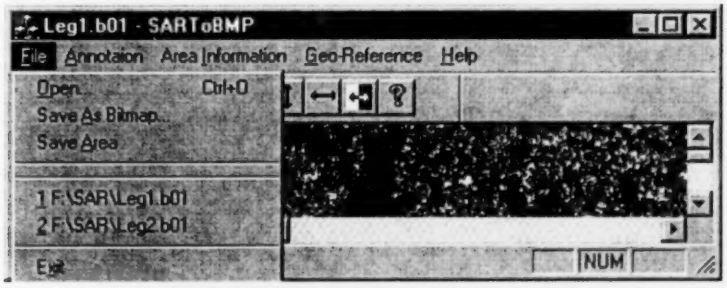
The simulator data is based on analysis of SAR imagery. In order for the analysis to occur the imagery needs to be put into a form that the MSP can read. The SARtoBMP.exe is a stand-alone application that is used to convert the SAR format data into Windows BMP format data. It also uses the geo-registration information in the SAR file to create a corresponding world file (Geo-Reference file). The SAR data is either 15 or 25 m per pixel. The width of a typical SAR Leg is 66 km or 100 km depending on the aforementioned pixel size. The SAR file data length is totally dependent on the flight line.

Initially these files resided on 8 mm Exabyte tape at Ice Centre. An archive utility called Exa2DOS was used to read these files from tape onto disk or CD-ROM. From here the SARtoBMP.exe utility normalizes the files into a Windows bitmap. It then reads the geo-coded information in the file and converts the file into a Windows bitmap. The program allows the user to flip and transpose the image in order to compensate for the direction of the airborne flight line.

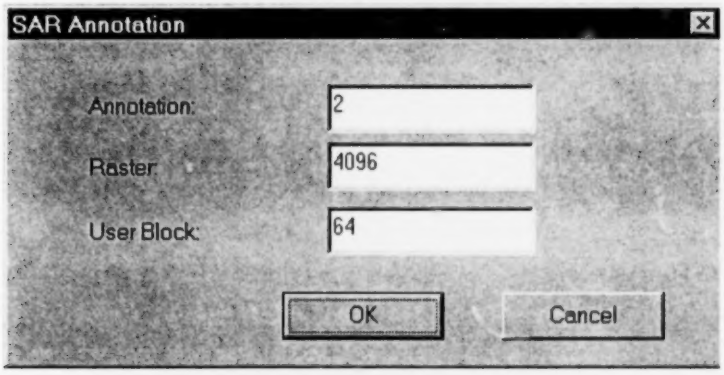
The SARToBmp.exe is capable of handling SAR files sizes of up to 600 Mbytes. To accomplish this without keeping the entire image all in RAM, the SARToBmp.exe application runs an image management scheme. The scheme swaps sections of the SAR image into memory so as to permit the operator to scroll through the image on a section by section basis. All the functions listed below pertain to the operations performed on the area of interest (AOI) in memory.

The SARToBmp.exe is able to perform the menu item functions listed below.

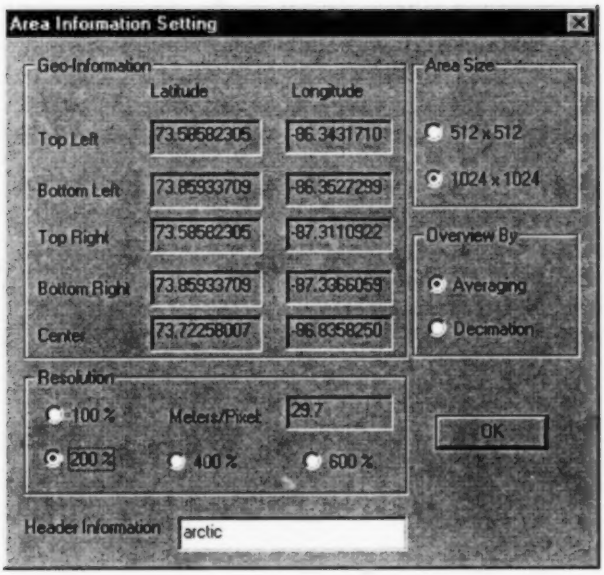
- **Open** a SAR File from the disk directory. When the user selects a SAR file, the Annotation dialog pops up so that the operator can modify the "line" parameters for the SAR file. If the entered parameters are incorrect, the image will be skewed on the screen. In this case, the operator can select "Annotation" from the menu to readjust the annotation parameters until the image is shown correctly on the screen.



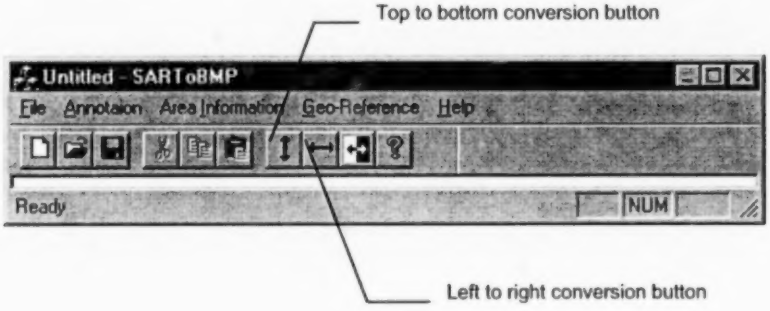
- The **Annotation** menu item can be used to pop up the Annotation Dialog so that the operator can modify the scan "line" parameters.



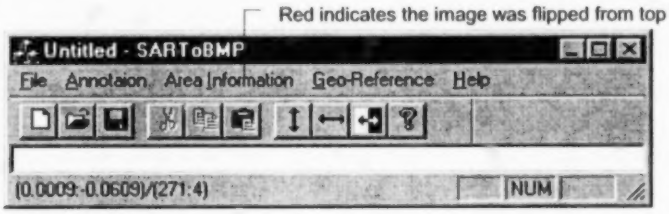
- **Area Information** is a menu item that is used to pop up the "Area Information Setting" dialog. This dialog is used to create "image segments" for use by the BBS simulator. The user can change or select an area of interest parameters. When the "Area Information Setting" dialog is displayed on the screen, the geo-reference information for the current area of interest is automatically shown on the dialog. If the user changes the resolution, size of the area, or the scrollbar position, the information is automatically updated. The header file is created according to this dialog's information.
    - Area Size: the size of a bitmap will be created (512x512 or 1024x1024).
    - Resolution: metres per pixel.
    - Overview By: the user can select a way to calculate a pixel's value when the resolution selected is bigger than 100%.



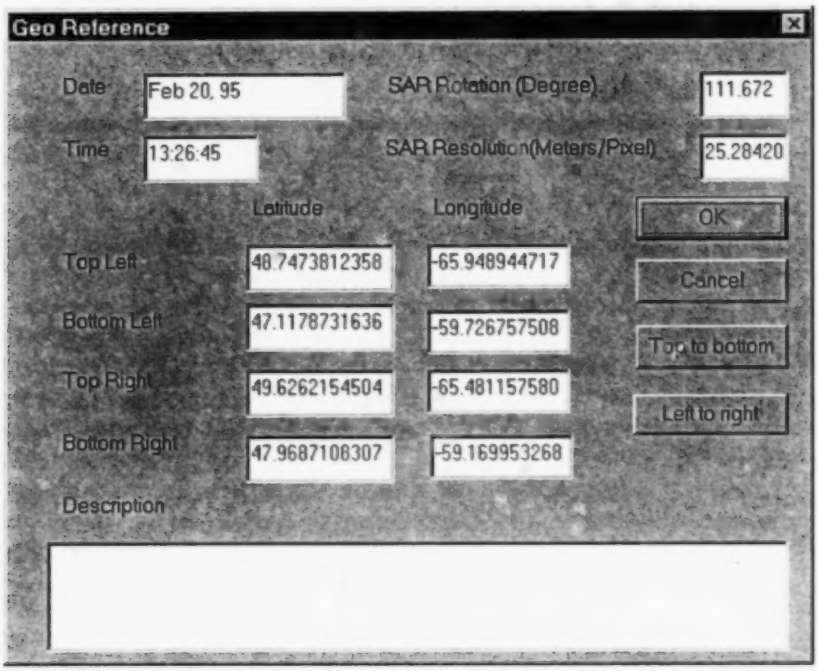
- **Top to bottom conversion** can be used to flip the image from top to bottom.



The color of the buttons for the "Top to bottom" and "Left to right" conversion is used to indicate whether the current image has been flipped or not. If the button is black, the image has not been flipped. If the color is red, the image has been flipped. For example, when the button for "top to bottom conversion" is clicked, the color of the button changes to red and the current image is flipped from top to bottom.


Red indicates the image was flipped from top to bottom
Untitled - SARToBMP
File Annotaion Area Information Geo-Reference Help
(0.0009:-0.0609)/(271:4)
NUM

- **Left to right conversion** is used to flip the image from left to right.
- **Geo-Reference** is used to pop up the Geo-reference dialog so that the operator can see or modify the geo-reference information for the current SAR image.



- **Save As Bitmap** pops up the Save As dialog so that the operator can select or enter a bitmap file name to be saved. The SARToBmp.exe coverts the current SAR file into a window bitmap file.
- **Save Area** pops up the Save As dialogs so that the operator can select or enter the bitmap file name to be saved. SARToBmp.exe converts an area of SAR file that has been defined in the "Area Information Setting" dialog to a bitmap file and creates the information in a header file.
- **Image Enhancement** is identical in operation to the function described in the SGM.exe section.

4.1.2. SARtoBMP Source Code Files

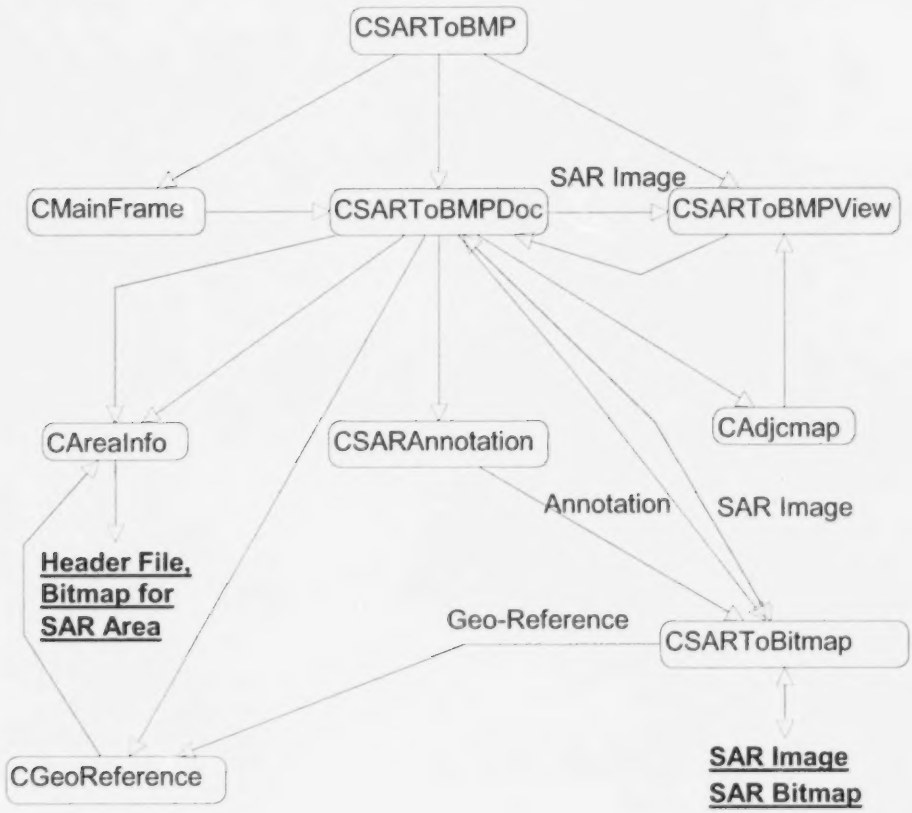
Table 4.1.2-1 lists the SARtoBMP source code files.

Table 4.1.2-1 SARtoBMP Source File Name List

SARToBMPDoc.cpp SARToBMPDoc.h	This file includes the CSARToBMPDoc class that is used to handle a document for SARToBMP.exe application.
SARToBMPView.cpp SARToBMPView.h	This file includes the CSARToBMPView class that is used to display SAR image on the screen.
SARToBMP.cpp SARToBMP.h	This file includes the CSARToBMPApp class that is used to construct CMainFrame, CSARToBMPDoc and CSARToBMPView classes.
MainFrm.cpp MainFrm.h	This file includes the CMainFrame class that is used to create the SARToBMP main frame, tool bar and status bar.
GeoReference.cpp GeoReference.h	See CGeoReference class in SGM.exe section.
SARToBitmap.cpp SARToBitmap.h	This file includes the CSARToBitmap class that is used to convert SAR image to Window bitmap image file.
AreaInfo.cpp AreaInfo.h	This file includes the CAreaInfo class that is used to supply an interface for the user to select parameters for saving a bitmap from an interesting area of SAR file. It also creates a header file according the parameters.
Adjcmap.cpp Adjcmap.h	The file includes the CAdjcmap that which is used to enhance the SAR image.
SARAnnotation.cpp SARAnnotation.h	The file includes the CSARAnnotation class that supplies an interface for the user to adjust the annotation's parameters for SAR file.

4.1.3. SARtoBMP Data Flow and Control

Figure 4.1.3-1 illustrates the relationship between the various objects together with the data and control flow.



where:

———	= Data Flow
----------	= Control Flow
[ ]	= Thread/Process/Class
ABC	= Data Store/Memory/Disk

Figure 4.1.3-1 SARtoBMP Data and Control Diagram

4.1.4. Scenario Generation Module

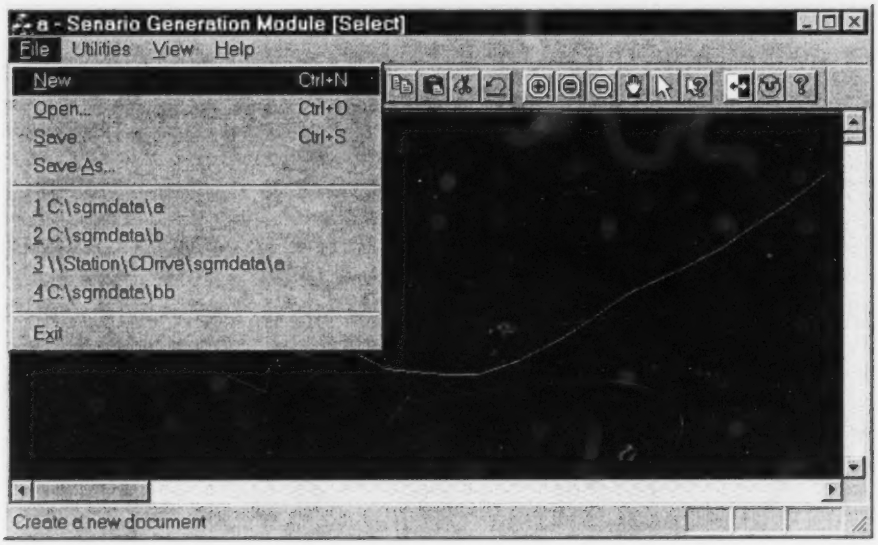
The SGM.exe is a stand-alone executable used to create and maintain the various scenario files for the MSP. The application reads large SAR Files in a piecemeal fashion and allows an operator to add layered topographical information onto the raster background. A separate header file provides the geo-registration information.

The Scenario Generation Module (SGM) is able to add, delete, or modify the vector data that corresponds directly with the feature data entered by the data analyst. This approach keeps the MSP self-contained and not reliant on the annotations methodology of a specific supplier or agency.

The SGM.exe allows an Ice Specialist to add detailed ice analysis information to the SAR data. The analysis data comprises polygons that contain references to ice type, thickness, and height. Because of the sheer volume of SAR data the SGM pages the data into memory limiting the actual visible memory to approximately 8 mega-bytes of data. The SGM.exe is capable of handling SAR file sizes of up to 600 Mbytes. To accomplish this without keeping the entire image in RAM, the SGM runs an image management scheme. The scheme swaps sections of the SAR image into memory so as to permit the user to pan through the image on a section-by-section basis. All the functions listed below pertain to the operations performed on the AOI in memory. If while drawing the feature, the region or feature outline exceeds a window width or height, the act of holding the mouse at the client window border causes the SAR image to scroll. If the feature is still larger than the area of interest, a new image segment is loaded into memory.

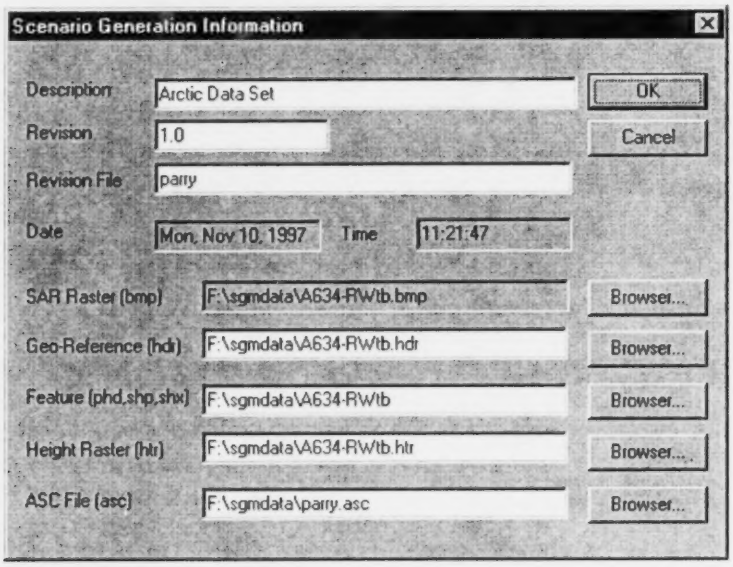
The following section describes the SGM user interface. Each of the menu item keywords and corresponding toolbar options is described below.

- Select **New** to create a new scenario generation file, with or without a corresponding SAR image file.



If all of the file name fields are empty, the base names of Geo-reference and Feature files are the same as for the SGM file name although with different file extensions.

If the geo-reference file was not found, the "Geo-Reference" Dialog will show on the screen and the user can enter Geo-Reference information via the dialog. This dialog is the same as in SARToBmp.exe program. SGM.exe reads the Geo-reference file that has been created from SARToBmp.exe.



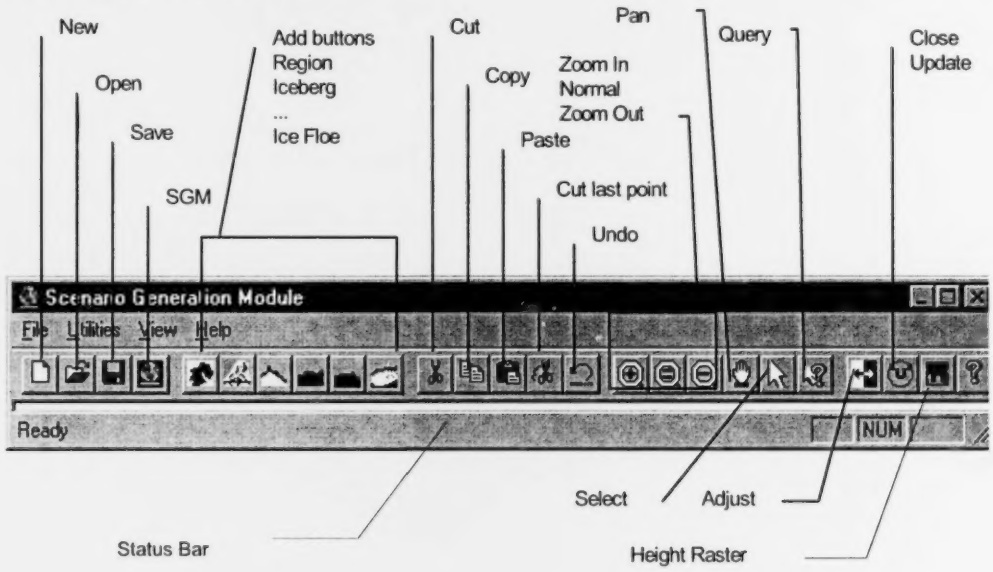
Note: If an existing Geo-reference file and Feature file were selected from the disk, the data is overwritten when the user saves the SGM file.

When the SGM file is loaded into the program, the status bar shows the cursor position in the world. From this point the user is able to:

- **Select** and **Open** a SGM File from the disk directory. The SAR data may exist on hard disk or on CD-ROM.
- Select **Save** to write the current SGM data to disk. If the file name was not entered then the Save As dialog pops up so that the user can select or enter the file name.
- Select **Save As** to change to save set name.
- **Add, Select, Edit, Move,** and **Delete** individual feature layers.
- **Add, Select, Edit,** and **Delete** a polygon vector data to the raster data or space.
- **Add, Select, Edit, Move,** and **Delete** Attribute Data. This is in the format of records that are tied to the individual layers and objects.

- Use the "**Add, Cut, Copy, Paste, Undo, Select,** and **Query**" buttons on the toolbar or items on the "Mode" menu to add or edit the feature data on the image. When a feature has been selected, the shape of the feature is highlighted in yellow. After a feature has been selected, the "Cut", "Copy", "Move", and "Query" functions are enabled. The "Paste" function is enabled after a "Cut" or "Copy" action has been executed. Once a feature is selected, the mouse is used to perform the "Move" function.

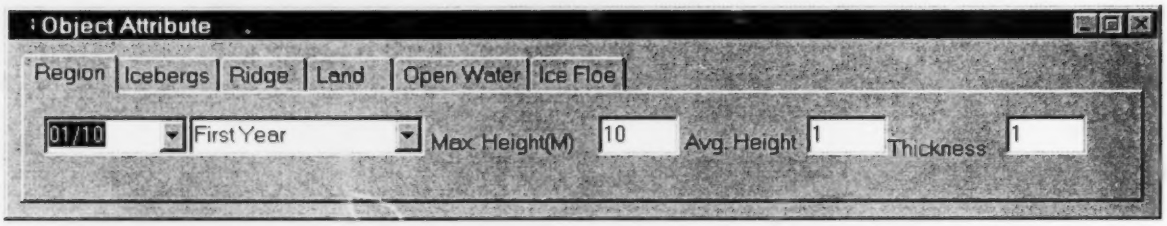
The SGM Toolbar and Status



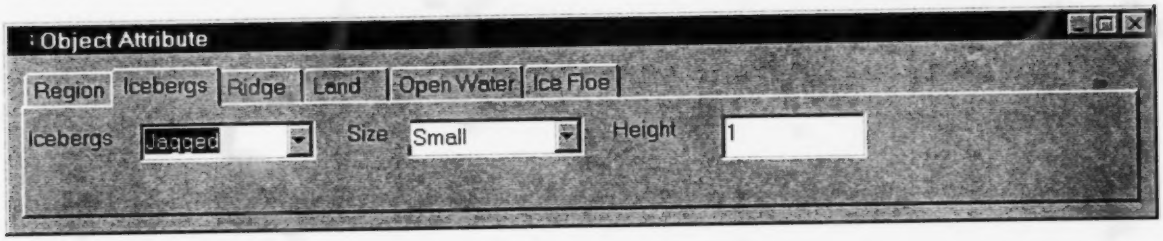
Bar:

- **Add Mode:** In "Add mode", the "Object Attribute" pops up so that the user can select the kind of features to add and enter the feature attributes. Once this is done the user moves the mouse and clicks mouse left button to effectively draw an outline of the feature boundary over the SAR image. This then completely defines the feature. All features are added with this same method. The figures that follow show the various types of features that can be added.

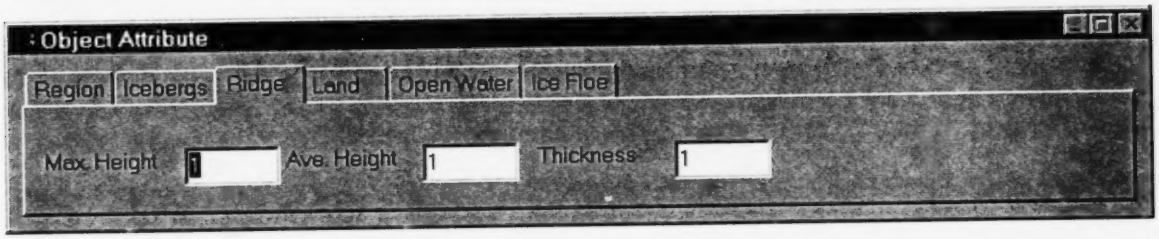
Add Region (Ice Concentration)



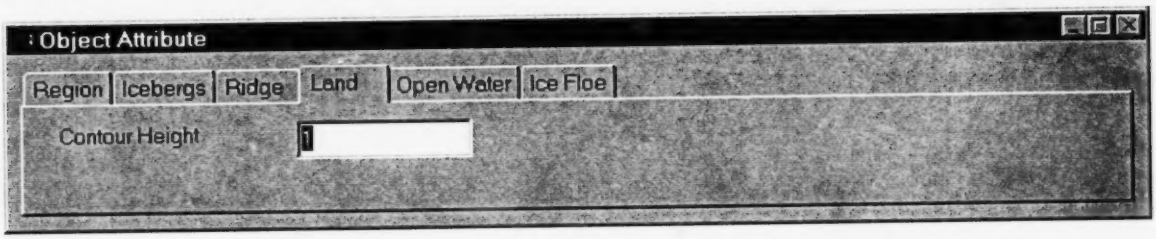
Add Iceberg



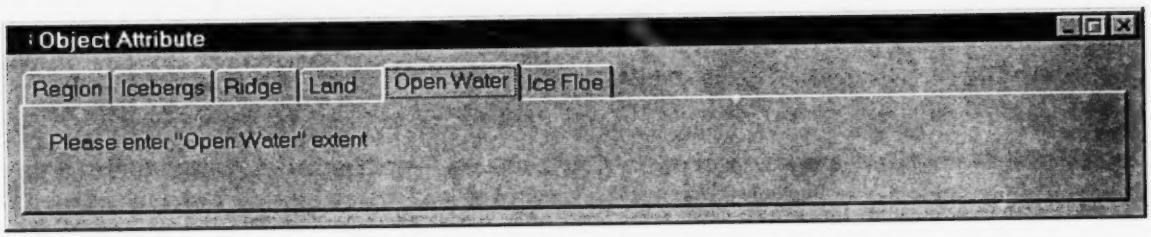
Add Ridge



Add Land



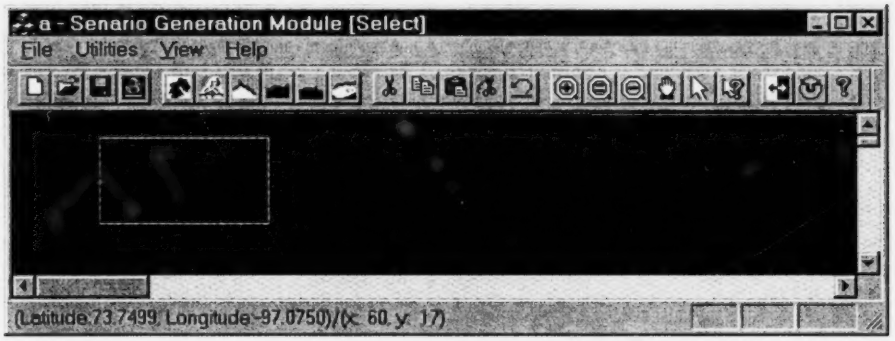
Add Open Water



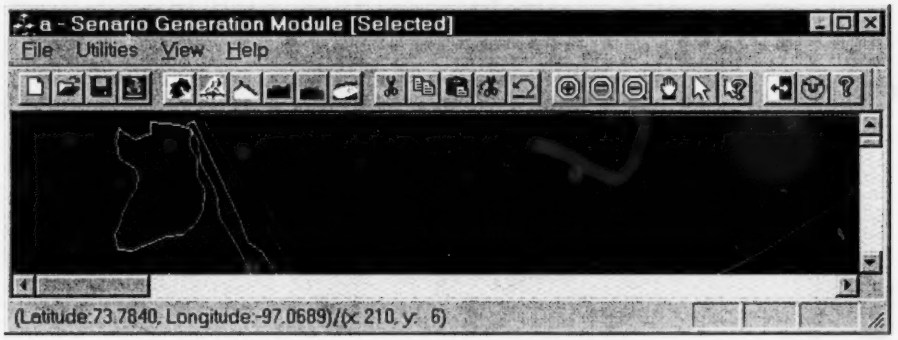
Add Ice Floe

- **Select (Entire Feature):** Use the mouse to select features on the image by pressing down the left button of the mouse, holding it, and moving the mouse. A box is shown on the screen. Any features contained in the box become selected.

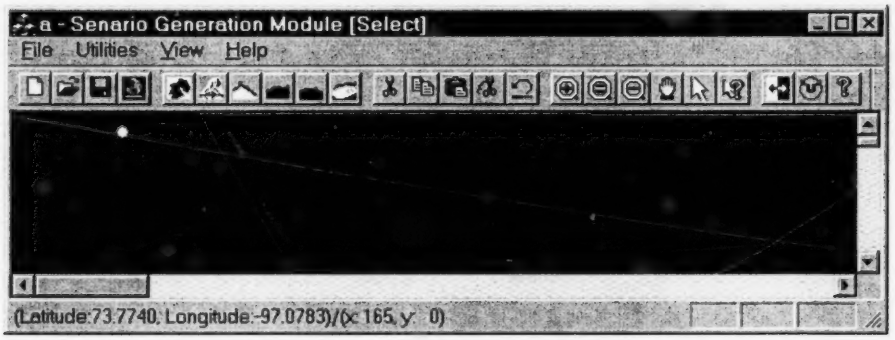
Select features:



Selected features are highlighted in yellow



- **Select (Feature Point/Vertex):** Use the mouse to point to a feature vertex. The feature vertex must be within four pixels of the mouse indicator. Once a vertex is selected the user can move or delete the vertex.

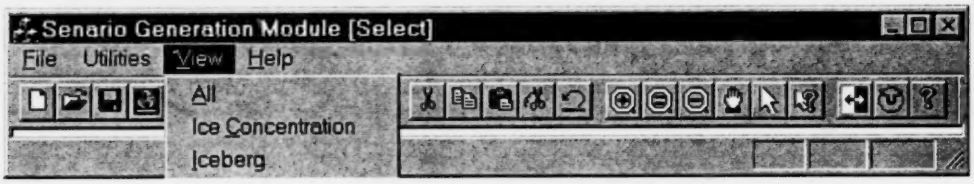


Selected point is highlighted with the yellow circle.

- **Move:** After a feature(s) or point has been selected the user can use the mouse to move the feature shape(s) to a new position. To do this click on the selected feature(s), then

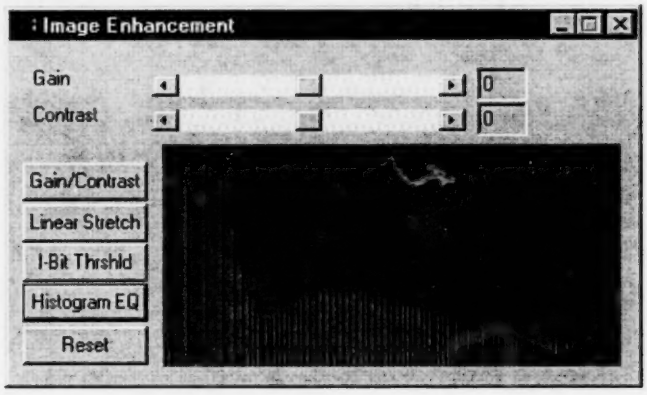
move the mouse with left button pressed to a new position, and finally release the left mouse button. The selected feature(s) is/are moved to the new position. For feature points select a point on the shape of a feature, move the mouse, and then release the button. The selected point is moved to a new position.

- **Turn On/Off** individual layers. To see an image with a particular layer, SGM.exe contains "View" menu to change the system view mode. If the view mode is on "View Iceberg", all of the icebergs on the image are shown on the screen. The remaining layers disappear from the screen.



- **Pan and Zoom down** the image bitmap and vectors. To pan through the SGM file select the hand icon in the main toolbar and move the mouse with the left button pressed. To zoom in, select the "+" magnifying glass icon and click the left mouse button for successive zooming. To zoom out select the "-" magnifying glass icon, again clicking the left mouse button for each successive zoom. To go back to normal scale select the "=" magnifying glass icon.

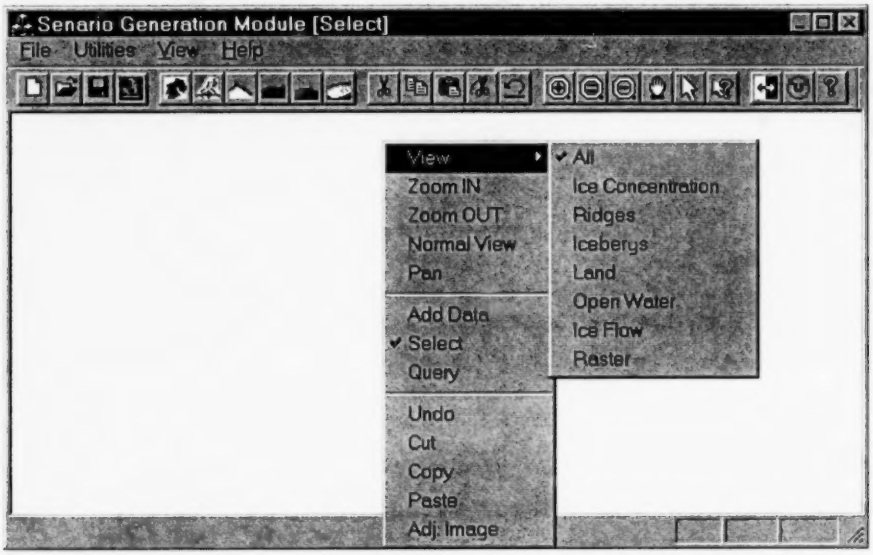
- **Adj. Image:** This item allows the user to enhance SAR image so that user can see the image more clearly. When the Adj. Image menu item or push button is selected, the Image Enhancement dialog pop-ups on the screen. The user can then change or adjust the image from the dialog.



- **Close or Update Button: Close**—This button is used to close a polygon that was drawn in the "Add" mode. **Update:** When an attribute of a feature has been modified in the "Modify" mode, this button is used assign or update the new data to the feature.
- **Undo:** By Clicking this button the user undoes the last action such as "delete, paste, move, or add".
- **Height Raster:** This button is used to generate a Height Raster file that will be used for the SRSM. When the button is selected, the Height Raster Processing dialog is shown on the

screen. The dialog indicates what percentage of the data has been processed. It also supplies an interface for the user to stop or abort the processing.

- **Color:** Color used to represent the meaning of a feature on the screen. The list below indicates the color to feature mapping.
  - Blue – Open Water
  - Cyan – Ridge
  - Green – Land
  - Purple – Region (Ice Concentration)
  - Red – Iceberg, Ice Floe
  - Yellow – Selected feature
- **Pop-up Mode menu:** To make the SGM.exe user-friendlier, a pop-up "Mode" menu appears when the user clicks the right button of the mouse. The user can then select a command from the menu. This pop-up menu acts an accelerator to the main menu.



- **Keyboard Control.** In order to accelerate data entry, there is a keyboard control function that enables the user to add or edit features as well as scroll the image by using the keyboard.

Key Name	**– Comments**
• Insert	– Add mode
• Delete	– Delete the selected features
• Home	– Scroll the current image to the top beginning position
• End	– Scroll the current image to the bottom position
• Page Up	– Scroll the image to the prior page
• Page Down	– Scroll the image to the next page

- Shift+Home – Scroll the image to the left side beginning position
- Shift+End – Scroll the image to the right side end position
- Up – Move the image up 8 pixels
- Down – Move the image down 8 pixels
- Left – Move the image left 8 pixels
- Right – Move the image right 8 pixels
- Center (keypad 5) – Move to the center position of the image
- F1 – Query the number of the features in the current file
- F2 – Generate a Height Raster file

4.1.5. SGM Source Code Files

The following table lists the SGM source file names.

Table 4.1.5-1 SGM Source File Name List

File Name	Description
Adjcmap.cpp Adjcmap.h	This file includes the CAdjcmap class that is used to adjust the image enhancement.
GeoReference.cpp GeoReference.h	This file includes the CGeoReference class that is used to pop up the Geo-reference Information dialog so that the user can modify the data.
getBmpSAR.cpp getBmpSAR.h	The file includes the CGetBmpSAR class used to get a part of an image from the SAR data file.
MainFrm.cpp MainFrm.h	The file includes the CMainFrame class used to create the SGM main frame.
PropertyPage1.cpp PropertyPage1.h	The file includes CPropertyPage (1-6) classes used to handle the feature attributes such as Region, Iceberg, Ridge, Land, Open Water, and Ice Floe.
PropertySheet.cpp PropertySheet.h	The file includes CPropertySheet class used to combine 6 PropertyPage classes.
PropertyFrame.cpp PropertyFrame.h	The file includes the CPropertyFrame class used to create the PropertySheet frame on the screen.
Senario.cpp Senario.h	The file includes the CScenario class used to supply the interface to the user so that the scenario generation data can be modified.
SGM.cpp SGM.h	This file includes the CSGMApp class used to construct the CMainframe, CSGMDoc and CSGMView classes.
SGMapi.cpp SGMapi.h	The file includes some public functions used to paint images on the screen.
SGMDoc.cpp SGMDoc.h	The file includes CSGMDoc class used to handle all the documents such as SGM file, feature files, geo-reference file.
SGMView.cpp SGMView.h	The file includes CSGMView class used to display image or feature data on the screen.
shape.cpp shape.h	The CShape class handles the features on the screen.
shapeFile.cpp shapeFile.h	The file includes CshapeFile class used to handle the save/read functions for feature data.
HeightRaster.cpp HeightRaster.h	The file includes the CheightRaster class used to generate the Height Raster file for the SRSM.exe application.
HeightProcessing.cpp HeightProcessing.h	The file includes the CheightProcessing class used to indicate the percent of the Height Raster file has been generated. It also supplies the "Close" button so that the user can stop the processing.
ImageProcessor.cpp ImageProcessor.h	The file includes the CimageProcessor class used for the user to preparate different SAR image files and Height Raster files based on the different resolutions selected from the user. The files will be used for SRSM.exe for different ranges selected by the user.

4.1.6. SGM Data Flow and Control

Figure 4.1.6-1 illustrates the relationship between the various objects together with the data and control flow of the SGM software.

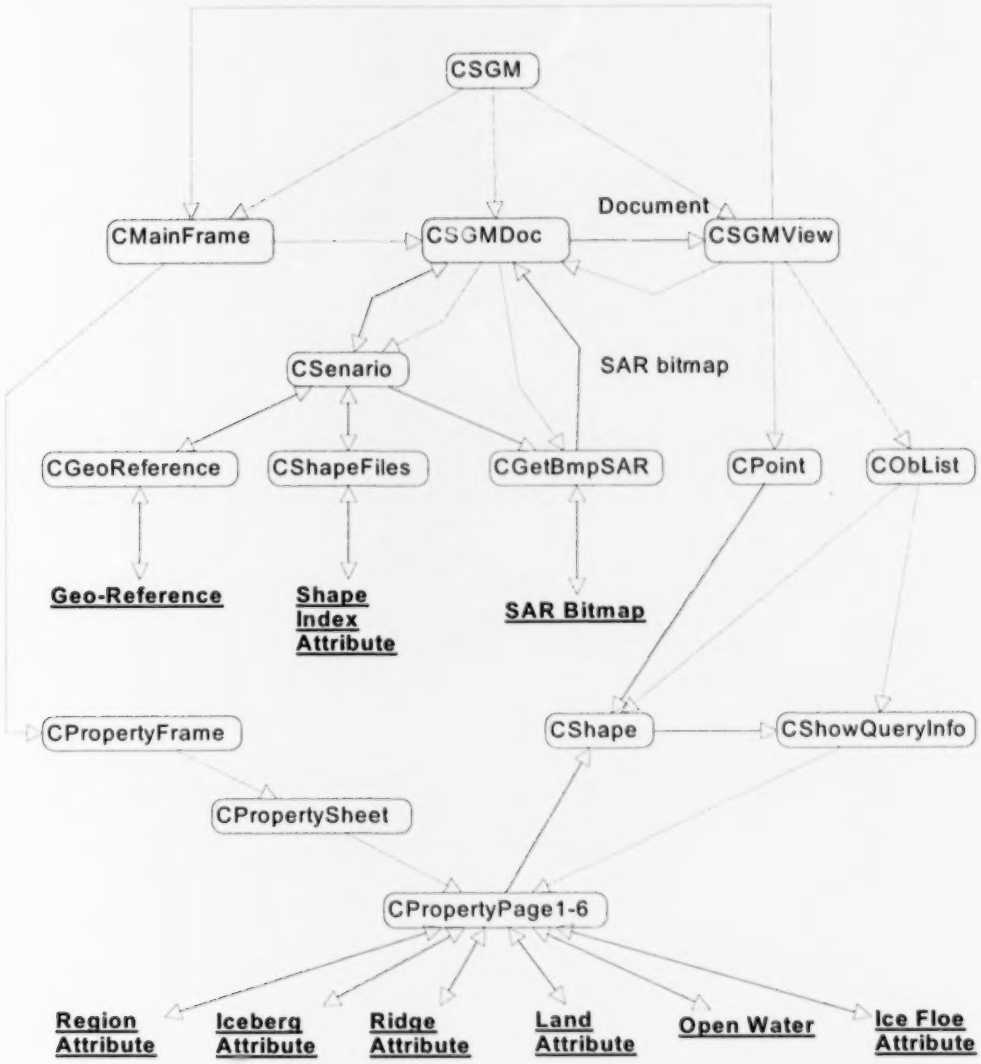


Figure 4.1.6-1 SGM Data and Control Diagram

4.1.7. Visuals Generation Module (VGM)

The VGM is responsible for generating the 3-D visual world. Once the user has finished using the SGM for analysis entry he/she exports the analysis to a feature ASCII file that the VGM imports. The VGM takes this file and creates a world file.

The VGM software is a procedure-based application. The following is a table of components that make up the VGM software. Included in the table is a description of each procedure.

Table 4.1.7-1 VGM Software Components

Procedure Name	Description
Form_Load	Sets the initial size of the main control window. This procedure is automatically run when the VGM application is started.
OK_Click	This procedure terminates and unloads the application. It is executed when the user "clicks" the OK button displayed on the main dialog.
Process_Click	Provides the main processing loop of the VGM.
XYMet	Converts Latitude and Longitude to X and Y meters from 0,0.
ReadFile	*Reads a line of text from a file that is terminated with a NULL character (""), and semi-colon (;) or a double forward slash (//).*
MakeFace	Writes a face record to the output file.
ClusterHeader	Writes a cluster header to the output file.
Getpoints	Returns the number of vertices in an ice feature.
Makepolyfile	Generates a *.poly file fir input to the "TRIANGLE" routine.
MakeVertices	Writes a vertices record to the output file.
MakeNormals	Writes a normal record to the output file.
MakeTextureList	Writes a texture record to the output file.
MakeCanvasList	Writes a canvas list record to the output file.

Process_Click

This procedure forms the main processing loop of the VGM module and is executed when the user clicks the "process" button". It opens the world file and for every feature in the input file it creates the object vertices, texture, and canvas list and makes the faces for all the triangles in the visual world.

Readfile

This routine reads an input file line at a time.

MakeFace

This routine writes a face record to the output file.

ClusterHeader

This routine writes a cluster header to the output file.

Getpoints

This function returns the number of points in a feature by reading it from the feature file.

MakePolyfile

This routine generates the *.poly file for each feature in the database. The format of the file is defined in the "Triangle" program documentation.

MakeVertices

This function reads an output *.node file generated by the triangulation routine and generates a vertices entry into the output file.

MakeTextureList

This procedure generates a texture list and texture u,v mapping based on an arbitrary control point outside a specific feature.

MakeCanvasLIst

This routine generates a canvas list depending on the specified ice type. Basically a switch case statement is used to determine the mapping between the ice texture bitmaps and the canvas ID number.

4.2. MSP/VSM Start-Up

4.2.1. MSP

Double clicking on the MSP icon in the MSP program folder starts the MSP.exe program. Other icons also exist in the folder. There is one for the Training Aids, one for the Scenario Generation Module (as a stand-alone application), and one for the actual simulator itself. Selecting the Training Aids will bring up the TAM as a stand-alone application complete with browser or help file engine. By selecting the SGM, a GIS style application opens allowing the operator to analyze and/or update the SAR feature files. Finally, by selecting the MSP icon the user starts all the processes associated with the MSP. This will include the MSP server. The VSM and SRSM are started as a separate autonomous process that is asynchronous to the MSP operation. Also the SRSM and the VSM are typically found on separate PCs due to their relative processing overheads.

Before the system is started a dialogue box (See Figure 4.2-1) is shown on the screen to enable the operator to set the initial conditions. This includes selecting:

- the scenario file,

as well as the setting initial conditions such as

- ship location or position,
- ship heading,
- ship speed,
- ship name,
- instructor and trainee names,
- initial throttle position,
- initial rudder angle,
- time of day conditions (day, night),
- wind velocity,
- wind heading,
- ocean velocity,
- ocean heading,
- visibly index (clear ...zero visibility),
- weather conditions,
- sky condition.

Once the initial parameters are entered and acknowledged they are used to initialize the system. The scenario file selection is used to select which set of SAR data to use in the simulator. The ship location is used to update the internal variables as well as select the AOI for the imagery. The ship's heading and speed is used to update internal MSP variables that will later be provided to the various subsystems. The ship name is used by the TSM to index into its ship type database. In this way the MSP need not know about the specific physical properties of a particular ship, but rather only the ship name. The instructor and trainee name form part of the recording session of the simulator. The initial throttle and rudder angles are synchronized in the MSP and passed to the TSM as requested. These parameters are updated as the trainee

modifies the rudder and throttle controls. Finally, the initial time of day and visibility index and weather information are stored in the MSP and given to the VSM when it requests it.

The instructor also has the ability to have the MSP record the current simulator inputs or to play back a different set based on previously recorded simulations. Also, once the system starts a specific scenario, it remains for the duration of the simulation or until the current one is terminated and a new one is chosen.

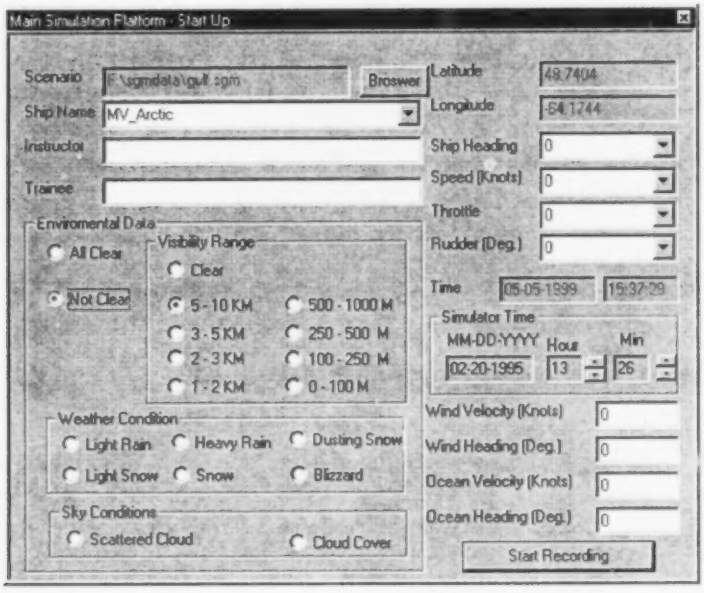


Figure 4.2.1-1 MSP Start-Up Dialogue

The instructor can use the browser button on the Start-Up Dialog to select a SGM file. When the button is clicked, the Open File Dialog is shown on the screen, and a SGM file can be read from a disk. Once the OK button at the Open File Dialog is clicked, the MSP displays a SAR image with all of the feature data so that the instructor can locate the ship's location and heading on the image.

Once the Run button on the Start-Up Dialog is selected, the MSP starts the network servers.

4.2.2. Data Flow and Control

Figure 4.2.2-1 represents the MSP Data and Control flow.

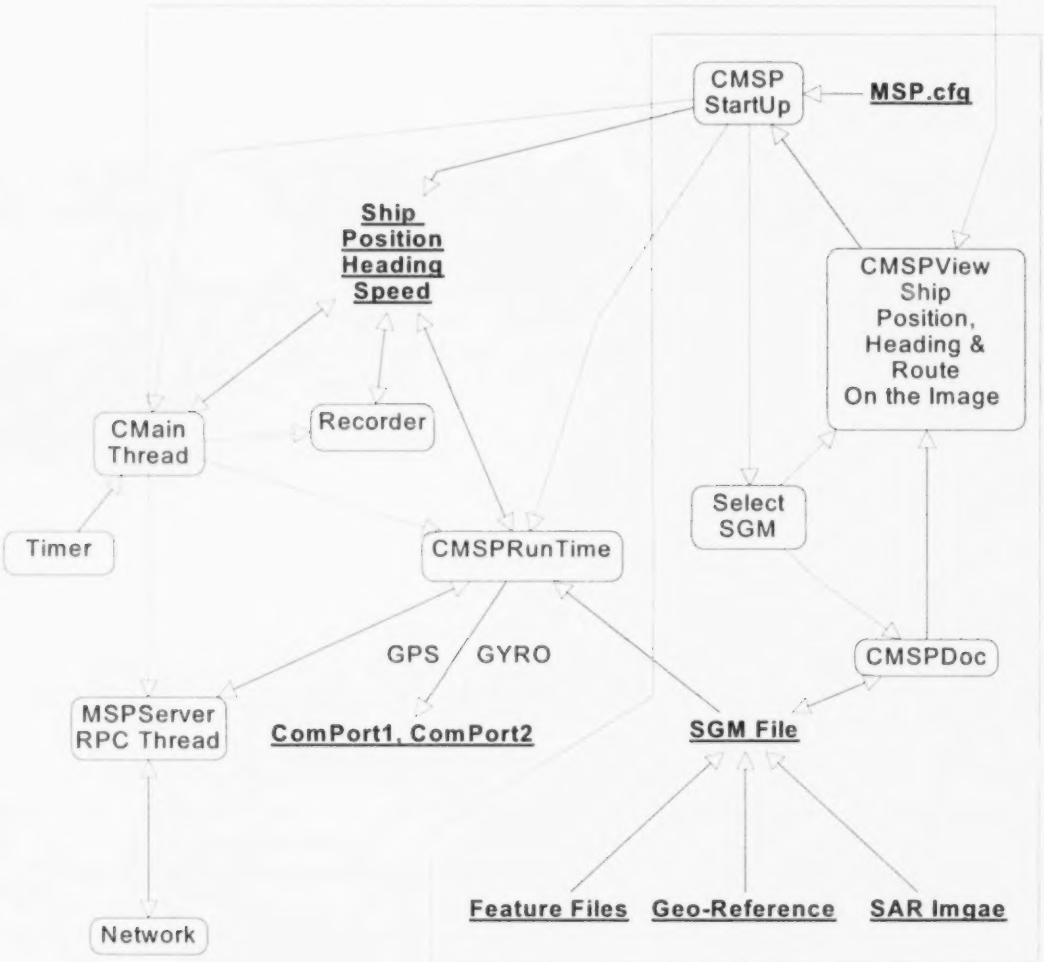


Figure 4.2.2-1 MSP Control and Data Flow

4.2.3. MSP User Interface

Figure 4.2.3-1 illustrates the MSP User Interface.

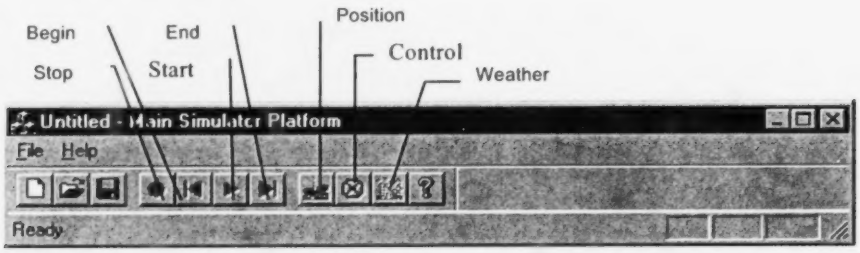


Figure 4.2.3-1 Main Simulator Platform GUI

A description of the Instructor commands indicated above are described below:

New - The menu item/button is used to create an MSP Session file that includes a path for the SGM file, the initial ship position and heading as well as initial environmental data.

Open - The menu item/button is used to open an existing MSP Session file.

Save - The menu item/button is used to save the current MSP Session file.

Save As- The menu command saves the MSP file as another file name.

Stop - The button is used to stop the current MSP session.

Begin - The button is used to move to the beginning of the track of the ship.

Start - The button is used to start the MSP session.

End - The button is used to move to the end of the track of the ship.

Position - The button is used to change the MSP mode so that the user can click a mouse to relocate the ship position on SAR image.

Control - The button is used to pop up the Ship Control Dialog so that the instructor can change the ship's operating controls (ex. Rudder, Throttle).

Weather - The button is used to pop up the Weather Control Dialog so that the instructor can change the current weather condition for the MSP.

The **Status Bar** uses the same functions as the SGM.exe to display the cursor position.

4.2.4. Ship Control Dialogs (Instructor and Student)

There are two Ship Control Dialogs. One is used for the instructor and is attached to the MSP.exe application. The other is a stand-alone application called "trainee.exe". Both can control the ship and both perform identical functions. The only difference is that the trainee has a "Help" button to invoke the training aid module (TAM). The user can change the Rudder and Throttle values via slider control in the one of the dialogs. The "Position" boxes show the current ship position as latitude and longitude. The "Heading" field indicates the current ship heading referenced to true north. Finally, the "Speed" entry indicates the current ship speed. Pressing the "C" button on each slider control causes the scrollbar to be centered thereby returning the corresponding control to its zero position.

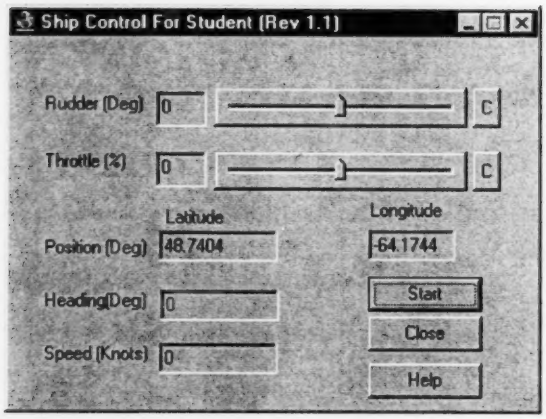


Figure 4.2.4-1 Ship Control Dialog

4.2.5. Weather Control Dialog

This dialog permits the instructor to change the environmental conditions of the simulator session.

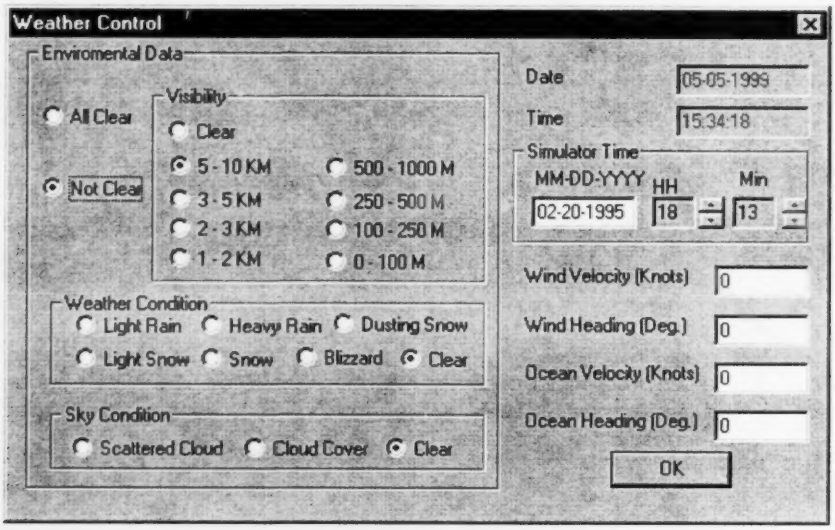


Figure 4.2.5-1 Weather Control Dialog

4.2.6. MSP Source Code Files

A list of the MSP source code files complete with a description of included objects is given in Table 4.2.6-1.

Table 4.2.6-1 MSP Source File Name List

File Name	Comments
MSPDoc.cpp MSPDoc.h	These files include the **CMSPDoc** class that is used to handle a document for MSP.exe application.
MSPStartUp.cpp MSPStartUp.h	These files contain the implementation of the **CMSPStartUp** class that is used to supply an interface for instructor to set the initial environment for MSP.exe to start, such as SGM file, ship condition, and weather conditions.
MSPView.cpp MSPView.h	These files include the **CMSPView** class that is used to display SAR image and feature data on the screen.
MSP.cpp MSP.h	These files include the **CMSPApp** class that is used to create the CmainFrame, CMSPDoc, and CMSPView classes.
ShipControl.cpp ShipControl.h	These files include the **CShipControl** class that is used to "drive" the ship.
WeatherControl.cpp WeatherControl.h	These files include the **CWeatherControl** class that is used to supply an interface for the instructor to change the current weather environment.
BDM.cpp BDM.h	These files include the **CBDM** class that is used to query data from SAR and feature files.
GPS.cpp GPS.h	These files include the **CGPS** class that is used to send the GPS data as an NMEA 0183 string to the communication port. (GLL format)
GYRO.cpp GYRO.h	These files include the **CGYRO** class that is used to send the GYRO data as an NMEA 0183 string to the communication port. (HDT format)
GPS_GYRO.cpp GPS_GYRO.h	These files include the **CGPS_GYRO** class that is used to contain CGPS & CGYRO classes.
ComPort.cpp ComPort.h	These files include the **CComPort** class that is used to communicate to the communication ports.
ComPortSetting.cpp ComPortSetting.h	These files include the **CComPortSetting** class that is used to initialize a data communication port.
MainFrm.cpp MainFrm.h	These files include the **CMainFrame** class that is used to create the main frame, tool bar, status bar and set the timer which is used to update the MSP.exe system. It is also used to supply a network interface to MSP client to query data.

GeoReference.cpp GeoReference.h	Refer to the **CGeoReference** class in the SGM.exe section.
GetBMPSAR.cpp GetBMPSAR.h	Refer to the **CGetBMPSAR** class in the SGM.exe section. Also includes a function to generate an overview of the SAR data.
Shape.cpp Shape.h	Refer to the **CShape** class in the SGM.exe section.
ShapesFile.cpp ShapesFile.h	Refer to the **CShapesFile** class in the SGM.exe section.
MSP_c.c	The MIDL compiler generates this file as C-language client stub file.
MSP_s.c	The MIDL compiler generates this file as C-language server stub file.
MSP.idl	The MSP.IDL file specifies a network contract between the client and server – that is, the IDL file specifies what is transmitted between client and server. Keeping this information distinct from the information about the operating environment makes the IDL file portable to other environments.
MSP.acf	The MSP.ACF specifies attributes that affect only local performance rather than the network contract.

4.2.7. VSM

When the WinVSM.exe application is started the 3-D world file is loaded into the graphics card memory. The ship's initial position and orientation in the 3-D world are determined by its initial latitude, longitude, and heading. The video is updated with every screen refresh. The VSM software is a set of Dynamic Link Libraries (DLL) and executable programs (EXE) that run in conjunction with the MSP software. The purpose of the VSM is to provide a module that displays a simulator 3-D view of the ship travelling in different ice conditions. The VSM displays a visual world pre-generated by VGM based on the analysis of Airborne SAR imagery. The VSM is a dialog window application that supplies an interface for the user to control the VSM view. The VSM gets the current ship information and weather information from the MSP.

4.2.8. Data Flow and Control

Figure 4.2.8-1 illustrates the basic data and control flow of the VSM software.

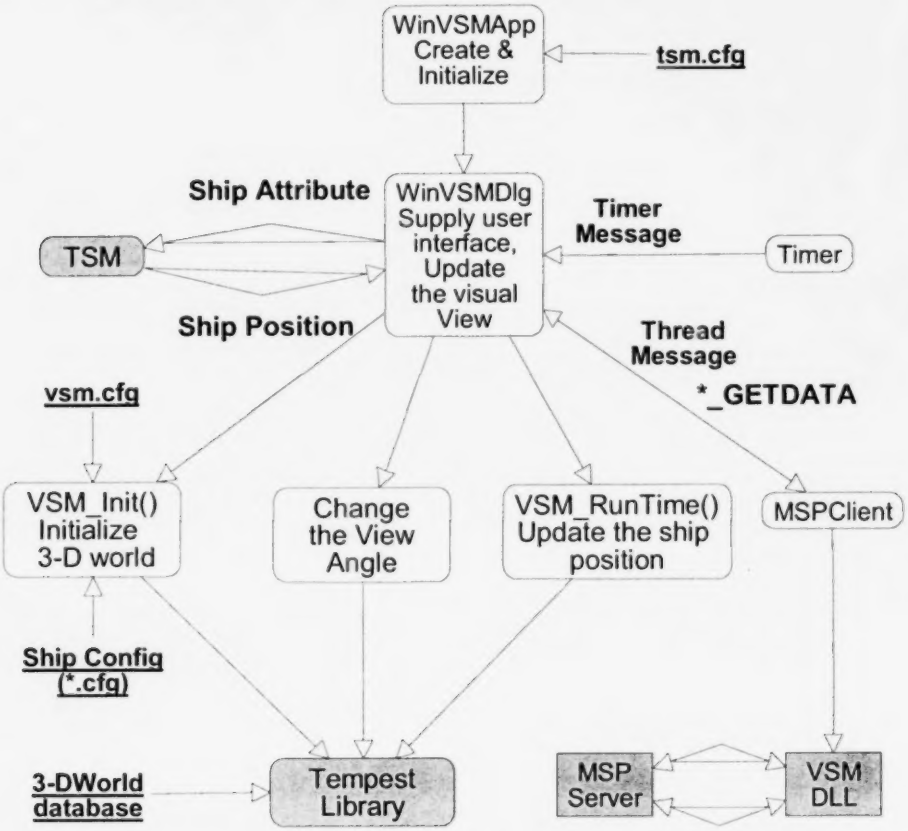
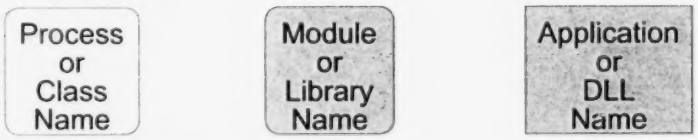


Figure 4.2.8-1 VSM Data Flow and Control Diagram

Note: see symbol legend below.

Where:



Data Flow = ———————————

Control Flow = -

Message Flow = ...

And a '*' is = WM_USER message

4.2.9. VSM User Interface

Figure 4.2.9-1 illustrates the VSM dialog for the user.

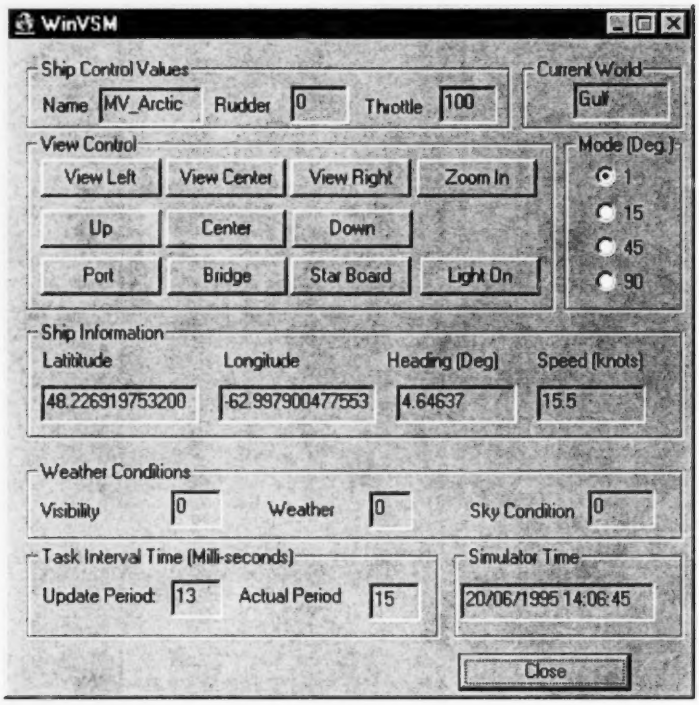


Figure 4.2.9-1 VSM User Interface

The **View Left**, **View Center,** and **View Right** buttons are used to change horizontal viewing direction relative to the bow of the ship. The **Mode** radio button is used to select the number of degrees the view direction changes for each left or right button click. The **Zoom In** toggle button provides a monocular view of the horizon.

The **Up** and **Down** buttons control the "bird's eye view" of the terrain, while **Center** brings the vertical view back to bridge height.

The **Port** and **Starboard** buttons change the user's "out of window" view to be set (off-center) to the left or right side of the ship respectively. Pushing the **Bridge** button brings the user back to the center of the bridge. Finally, **Light On** is a toggle button used to control the simulated spotlights.

4.2.10. VSM Source Code Files

A list of the VSM source code files and a brief description is given in Table 4.2.10-1.

Table 4.2.10-1 VSM Source File Name List

File Name	Description
base_parser.cpp base_parser.h	Contains the **Base_parser** class used to support basic functions for all flat file parsing.
Cartesian.h	Defines the Cartesian class used to do conversion between the Latitude/Longitude and the Cartesian.
Console.cpp console.h	This file supplies functions for a file reading or writing.
Constants.h	Defines the unit conversion constants.
filter.cpp filter.h	Contains the **CFilter** class used to filter the changing of the rudder or throttle values so that the changing of rudder throttle can not be jumped from one step to another step.
Latlong.cpp Latlong.h	Contains the **Latlong** class that is used to support functions for the conversion between the Latitude and Longitude and the Cartesian X and Y coordinates.
Mspcallsdef.h	Defines the constants and structures that are used for the network data communication.
MSPClient.cpp MSPClient.h	Contains the **CMSPClient** class that is used to support functions for the network communication with MSP.
Ocean_dyn.cpp Ocean_dyn.h	Contains the **Ocean_dyn** class that is used to compute the new position of the ship based on the ship property and the current ice environment.
Ocean_dyn_parser.cpp ocean_dyn_parser.h	Contains the **Ocean_dyn_parser** class that is used to parse the ship property from a sip configuration file.
Positional.cpp Positional.h	Contains the **Positional** class that is used to determine the ship position.
Resource.h	Defines the resource data.
StdAfx.cpp StdAfx.h	Contains the source file that includes just the standard.
Sun.cpp Sun.h	Contains a function that is used to compute the lighting of the sun based on the simulated time.
timer.cpp timer.h	Contains the **Timer** class that is used to perform basic timing operations.
tsm_config_parser.cpp TSM_config_parser.h	Contains the **TSM_Cfg_Parser** class that is used to parse the TSM configuration file.
TSMClient.cpp TSMClient.h	Contains the **CTSMClient** class that is used to support functions for the network communication with MSP.
TSMDll.h	Contains the interface to call the TSM.Dll functions.
vsm.cpp vsm.h	Contains the global functions that are used to load a 3-D world, a ship model and update the 3-D view on the screen.
vsm_struct.h	Contains the structure definitions for the vsm.cpp file.
VSMDll.h	Contains the interface to call the VSM.Dll functions.
WinVSM.cpp WinVSM.h	Contains the **CWinVSM** class that is used to initial the application.
WinVSMDlg.cpp WinVSMDlg.h	Contains the **CWinVSMDlg** class that is used to support an interface for the user to control the VSM view.

4.2.11. SRSM and SRSM Server

The next server to be started after the MSP is the SRSM. The SRSM server queries the MSP for raster bitmaps that are 1.5 larger that the range setting of the radar. This data is loaded in to the SRSM server's memory. The SRSM server is queried by a client application such as the Ice Navigation System or ECDIS.

4.2.12. MSP – GPS GYRO

The MSP also generates the NMEA GPS and gyro messages for the Ice Navigation and ECDIS systems.

4.2.13. Transit Simulator Module

In its present implementation the Transit Simulator is embedded in the VSM. This was done mainly for performance reasons. When the TSM was run "across" the network there was jitter on the screen. To alleviate this, the TSM was moved "closer" to the VSM. The VSM application is the only time-critical application in the system. In order to have a smooth flicker free image, the frame update rate must be at least 20 frames per second. If the same technology is to be used in immersed virtual reality, then at least 60 frames per second are required in order that the operator not experience motion sickness.

The TSM is currently a shell program that uses simple first order filters to facilitate damped motion and provides the resultant heading and position of the ship. This is just an interim solution until more suitable algorithms are implemented in the future. What remains in question at this time is the data required for the new TSM. Namely, is everything that is going to be needed currently inside the MSP database, or will new data need to be added?

4.2.14. Navigational Aids – IceNav and ECDIS

In order to utilize the Ice Navigation and/or ECDIS systems they need to be started. The principle use of the MSP, by the ECDIS system, is for radar simulation. Because the SRSM respects the MRI DLL interface specification, any client that works with a MRI should in fact work with the SRSM. This is described in more detail in the SRSM final report.

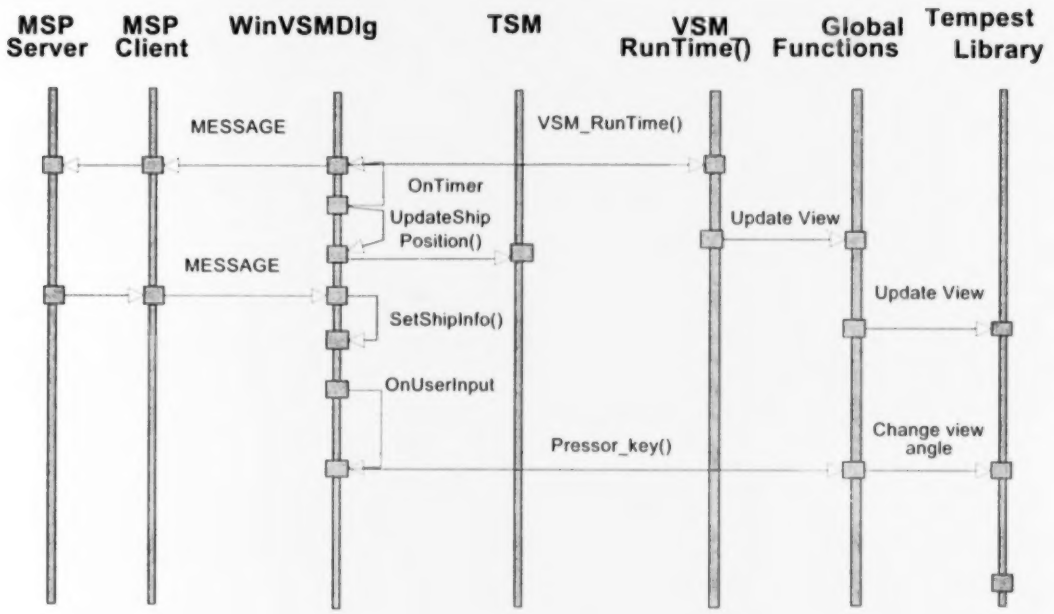
The Ice Navigation System is also supported by GPS and GYPO NMEA messages that are provided by the MSP.

4.3. MSP/VSM Run Time

4.3.1. VSM Run Time Scenario

Figure 4.3.1-1 represents the scenario diagram of the VSM at run time. At initialization, WinVSMDlg creates a timer whose time period is defined in the vsm.cfg file.

This timer causes the operating system to periodically send a WM_TIMER message to the WinVSMDlg thread. For every timer message received, WinVSMDlg sends a "MESSAGE" to the MSP Client. Once the MSP Client gets its data from the host (MSP Server) it signals the WinVSMDlg thread of its completion by re-sending the same "MESSAGE". The ship position, heading, and weather information are updated at this time.



NOTE:

MESSAGE: WM_USER_GETDATA

OnUserInput: When the user clicks on a button, the response function will be called.

Figure 4.3.1-1 VSM Run Time Scenario Diagram

4.3.2. MSP – Server

The MSP Server described in more detail later in this document supplies the network interface to all the components of the MSP.

4.3.3. MSP – Base Data Management

The MSP BDM is the executable code that actually performs the queries of the MSP database.

4.3.4. GPS and GYRO Message Generation

The MSP supplies NMEA 0183 interface strings to both the Ice Navigation System (INS) and the Electronic Chart and Display System (ECDIS). The position of the ship and its heading will be made via two messages.

The initial data for the communication settings is read and decoded from the MSP.cfg file and includes the following fields:

- COM Port name
- Baudrate
- Number of data bits
- Parity Even/Odd/None
- Number of stop bits
- Xon/Xoff S/W protocol

The format of the GPS message is as follows:

$--GLL, llll.ll, a, yyyy.yy, a, hhmmss.ss, A*hh<CR><LF> where

- llll.ll, represents Latitude
- a, represents N/S
- yyyy.yy. represents Longitude
- a represents E/W
- hhmmss.ss represents time
- A* represents status. A represents the data valid
- hh - checksum field

The format of the GYRO heading message is as follows:

$--HDT, x.x, T*hh<CR><LF> where

- x.x, T will represent the heading, in degrees relative to true north
- hh will represent the checksum field

4.3.5. MSP – Downlink

The MSP comprises an off-the-shelf program called World Group Manager (by Galacticomm). The program makes available a BBS style user interface that permits the student to practice using a land-based data provider.

5. MSPServer

The **MSPServer** supplies a network interface to MSP clients so that they can query data from MSP. It will use Microsoft RPCs to perform the data communication between the MSP Server and the MSP clients.

5.1. MSP Inter-process Communications

Figure 5.1-1 shows the inter-process communications between the MSP Server and its clients. Microsoft RPCs are used to implement the communication.

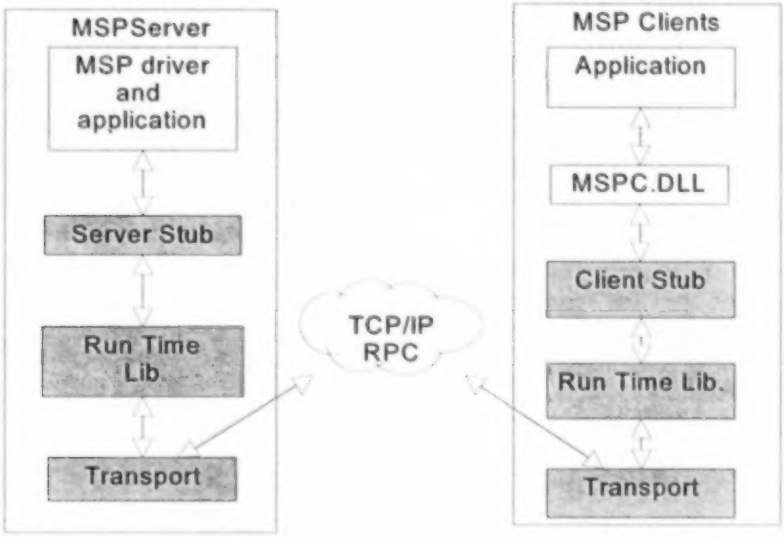


Figure 5.1-1 MSP Server and Client(s) Communication Architecture

5.2. MSPServer Functions

Figure 5.2-1 represents the function calls between the MSP server and its clients. Each client subsystem has it own set of interface calls.

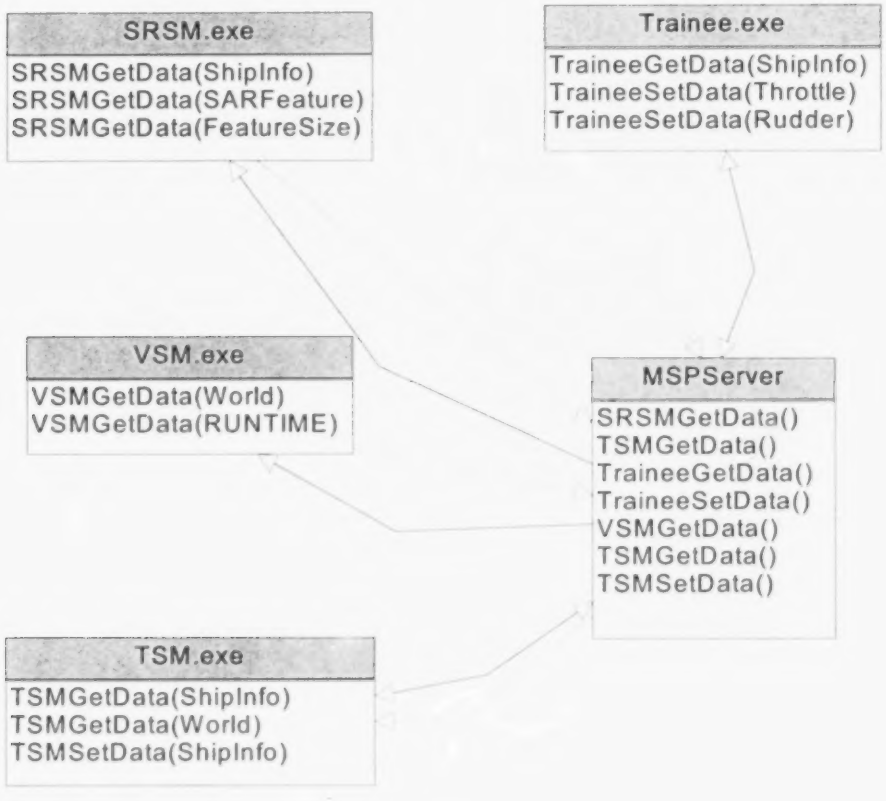


Figure 5.2-1 MSP Function Calls Between MSP Server and MSP Clients

The MSPServer call entries are described below.

SRSMGetData(UINT SHIPINFO, double range, void *DataBuf, void *SARBuf, void *Feature);

/*

This function provides the SRSM with ship information such as ship position and ship heading. The ship data is stored in **DataBuf** whose structure is of type **SHIPATTRIBUTE.**

*/

SRSMGetData(UINT FEATURESIZE , double Range, void *DataBuf, void *SARBuf, void *Feature);

/*

This function returns the size of the feature data to be located in memory by the caller. **Range** is an input value used to indicate extent of the query. The updated void **DataBuf** pointer is cast as a pointer to an object of type double so that the SRSM can determine the number of bytes to allocate.

*/

SRSMGetData(UINT SARFEATURE , double range, void *DataBuf, void *SARBuf, void *Feature);

/*

This function returns the SAR and feature data to the calling program. **SARBuf** is used to store the SAR image and **Feature** will be used to store feature data. **DataBuf** is used to store SAR resolution and rotation relative to true north.

The structure of **DataBuf** is

typedef struct SAR_INFO

{

float resolution,

float heading

};

*/

VSMGetData(UINT WORLD, void *FeatureData, void *NULL)

/*

This function returns the feature data in ASCII format.

*/

VSMGetData(UINT VSMRUNTIME, void * shipInfo, void *weatherInfo);

/*

This function is used by the VSM.exe to get the current ship and weather information from the MSP. The parameter **shipInfo** is stored as a SHIPATTRIBUTE structure. The **weatherInfo** parameter is stored as a WEATHERINFO structure.

*/

ShipControlSetData(UINT THROTTLE, float NewThrottle);

/*

This function changes the throttle of the ship.

*/

ShipControlSetData(UINT RUDDER, float NewRudder);

/*

This function changes the rudder of the ship.

*/

TSMGetData(UINT WORLD, void *FeatureData)

/*

This function returns the feature data in ASCII format.

*/

TSMGetData(UINT SHIPINFO, void * shipInfo);

/*

This function is used by the TSM.exe to get the current ship and weather information from MSP. The **shipInfo** parameter will be stored as SHIPATTRIBUTE structure.

*/

TSMSetData(UINT SHIPINFO, void * shipInfo);

/*

This function will be used by the TSM.exe to set the ship's speed and heading based on the existing ice conditions and ship performance parameters (i.e. horsepower). The shipInfo will be stored as SHIPATTRIBUTE structure.

*/

5.3. MSPServer and MSPClient.DLL

Figure 5.3-1 illustrates graphically the files and steps required to build MSPServer and MSPClient.DLL.

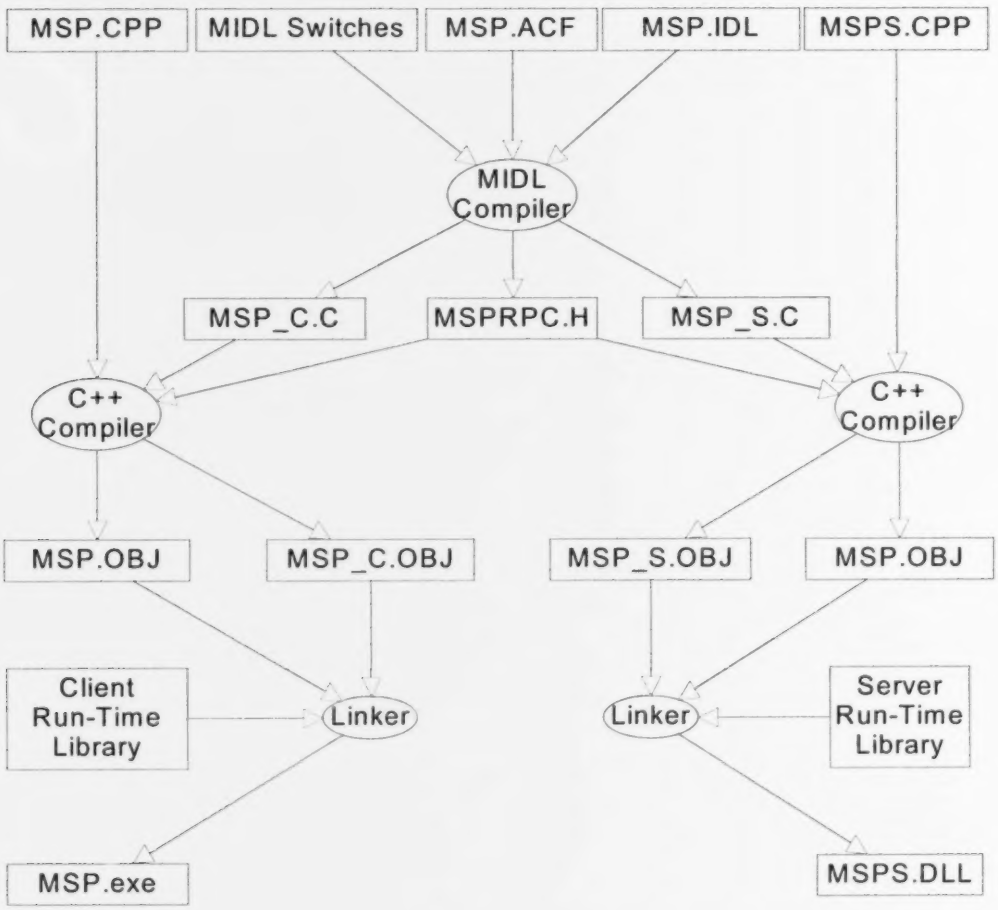


Figure 5.3-1 MSP Function Calls Between MSP Server and MSP Clients

The MSP.IDL file will be built by using the Microsoft Interface Definition Language Compiler. It includes the following functions.

```
[ uuid(53fc0226-f9fc-11d0-82bb-0060b01a28b0),
  version(3.1),
  pointer_default(unique)
] interface rpcGetData
{
void SRSMGetData([in] int type, [in] double range,
```

```
        [in] long length1,  [out, size_is(length1)]  char DataBuf[*],
        [in] long length2,  [out, size_is(length2)]  char SARBuf[*],
        [in] long length3,  [out, size_is(length3)]  char FeatureBuf[*]          );
void VSMGetData([in] int type, [in] long length1,   [out, size_is(length1)]  char data1[*],
        [in] long length2,   [out, size_is(length2)]  char data2[*]);
void TSMGetData([in] long plength,    [out, size_is(plength)]  char data[*]  );
void TSMSetData([in] long plength,   [in, size_is(plength)]  char data[*]);
void TraineeGetData([in] long plength,   [out, size_is(plength)]  char data[*]);
void TraineeSetData([in] int type,    [in]  float setValue);
void Shutdown(void);
}
```

5.4. MSPServer Implementation

Figure 5.4-1 represents how the MSP interchanges data and synchronizes with its clients. The design approach is to give unique RPC entries for each client. This prevents one client RPC from blocking a RPC from another client. When the RPCThread on the MSP Server is called from a client, it posts a message to its MainThread to "getdata". It then waits for another call from another client. When the MainThread receives the "getdata" message, it queries the BDM for data. When the query is finished, an event will be reset in order to "signal" the RPCThread to resume execution and return the data to the client.

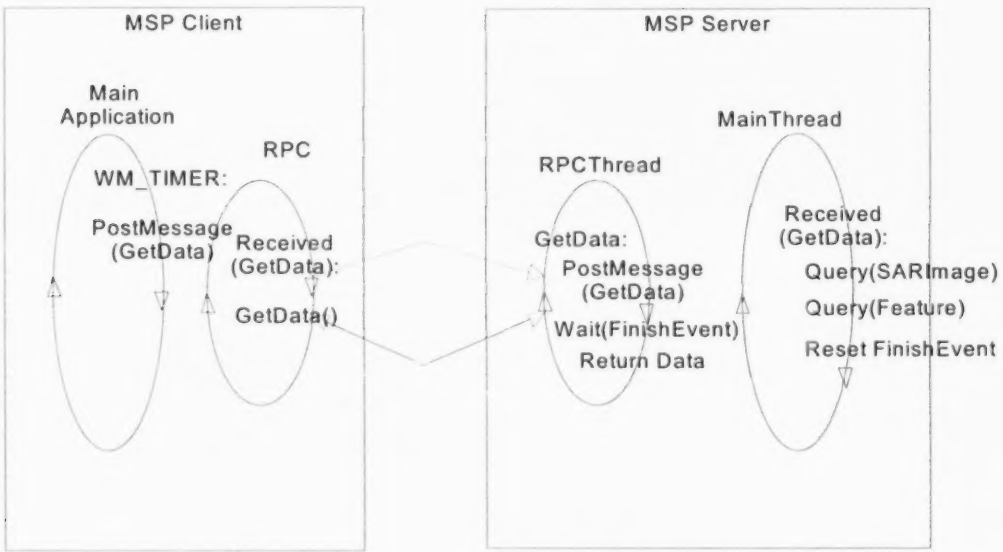


Figure 5.4-1 MSP Client Server Implementation

6. Typical Operational Use

The following section outlines the typical operational use of the simulator. To a navigator, piloting in ice is composed of utilizing several different types of information. This information comes from:

1. Out of Window Bridge Views – These views are typically in the range of 0 ~ 1.5 Nautical Miles (NM) in and around the ship. An operator uses either his eyes or binoculars to visually classify the ice.

2. Marine Radar Views – Typical radar range settings are usually in the range of 1—12 NM. The marine radar allows the operator to "see" further but also indicates ice obstacles in fog or night conditions. Certain types of radar also permit the operator to characterize the ice as being old or new. Old ice, unlike new ice, is as strong as steel and can tear a hole into the hull of a ship if hit too quickly.

3. Ice Navigation Tools Views – (12NM ~ 100NM) This data is typically satellite imagery that has been processed and downloaded to the ship via a communications satellite. Because the data is geo-coded the operator can use the Ice Navigation Tools to position the vessel in the imagery and plan routes accordingly. These tools provide a longer-range view of the ice conditions and allow the navigator to plan his route well in advance.

4. Navigation Instrumentation (GPS/Gyro Compass/Throttle/Rudder) – Provide the ship's direction and location as well as the ability to input steering and throttle control.

When a navigator wishes to navigate in ice-infested waters he plans a route using the Ice Navigation Tools. This allows him to determine his long-range route. Once under way and entering ice-covered waters, he uses the marine radar and his binoculars to determine his immediate steering tactics. His recognition of the ice types and the search for open water via the marine radar permit him to safely transverse the ice fields. Because the environmental conditions are constantly changing he may need to refresh his strategic view by getting an updated long-range view of the ice and modifying his route accordingly.

Figure 6-1 displays the ship's initial position in the MSP. It is important to observe the strong correlation of the information between the IceNav Image, the simulated marine radar image, and the "out of window display". The resolution of the imagery in all cases has been reduced in order to fit on the pages.

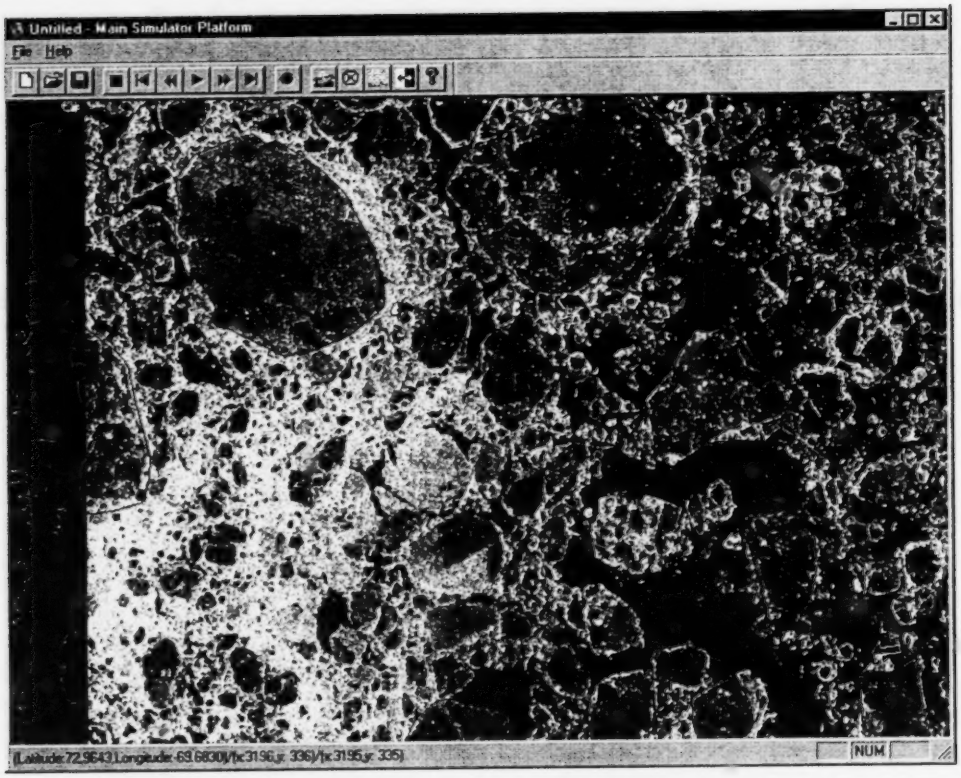


Figure 6-1 MSP Typical Use Example

6.1. Via Satellite and Remotely Sensed Data

Although not limited to a particular use it is envisioned that typically a user will be required to plan a route based on imagery received from a satellite. This will require that a route be planned and can be accomplished with an Ice Navigation Tool. This forms the strategic plan.

Images will be "downloaded" from the simulated BBS. These images are generated ahead of time, are geo-coded and compressed and received in much the same way as they would be from a BBS. Figure illustrates a downloaded image segment.

Once the route is planned the instructor can place the ship at various points along the route. The student is then required to use the radar and the out of window view to perform his tactical maneuvering. The operator can place the ship at various points along the planned route and navigate tactically through the ice. This can be done with the aid of the radar and the "out of window" view. One convenient feature of the simulator is the ability to change the time of day. The same area can rerun either in bright daylight, at night, or in varying degrees of visibility.

Figure 6.1-1 illustrates an example of a BBS style image that can be used to plan the initial route.

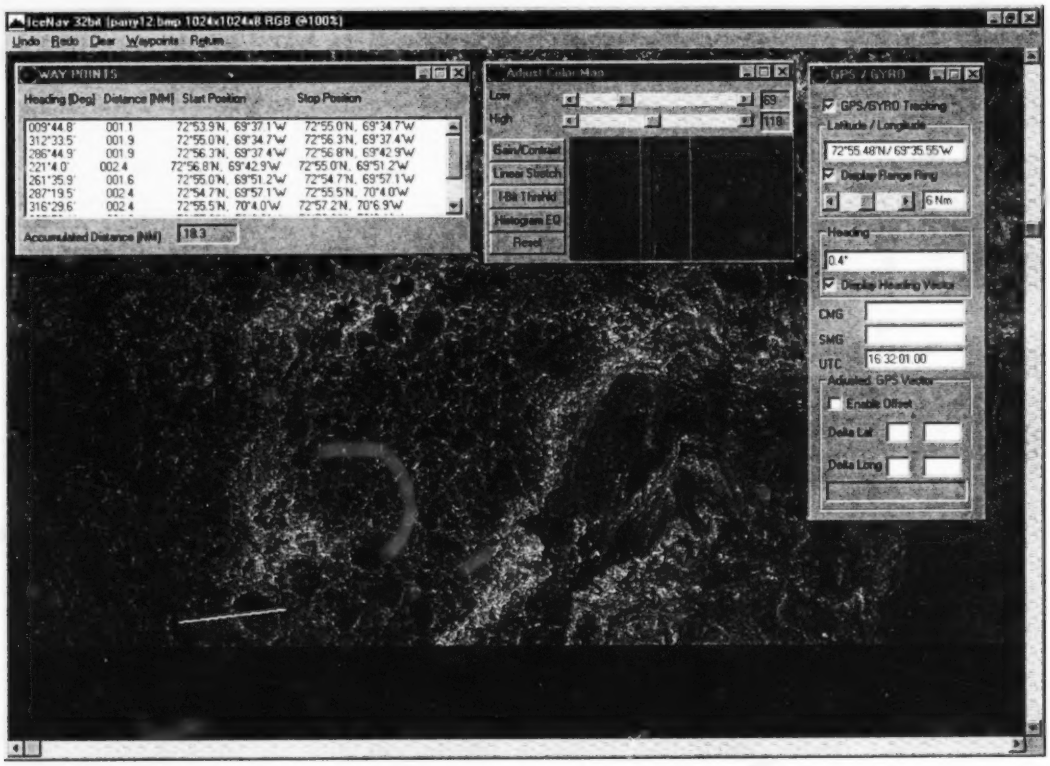


Figure 6.1-1 IceNav32 Route Planning Example

6.2. Via Marine Radar

Figure 6.2-1 illustrates the corresponding radar view of the ship's position as indicated in Figure 6.1-1.

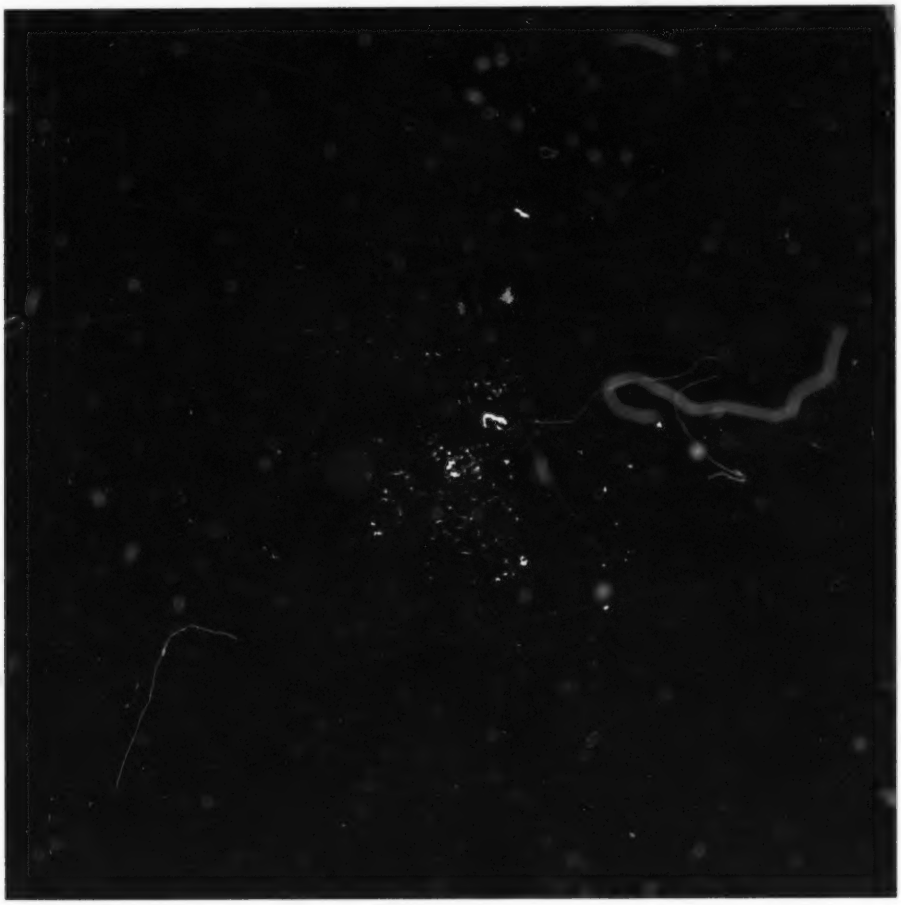


Figure 6.2-1 SRSM image #1

Note: The radar image range is set to 3 nm and OSCFAR is set to OFF.

Figure 6.2-2 illustrates the same are but with different image settings.

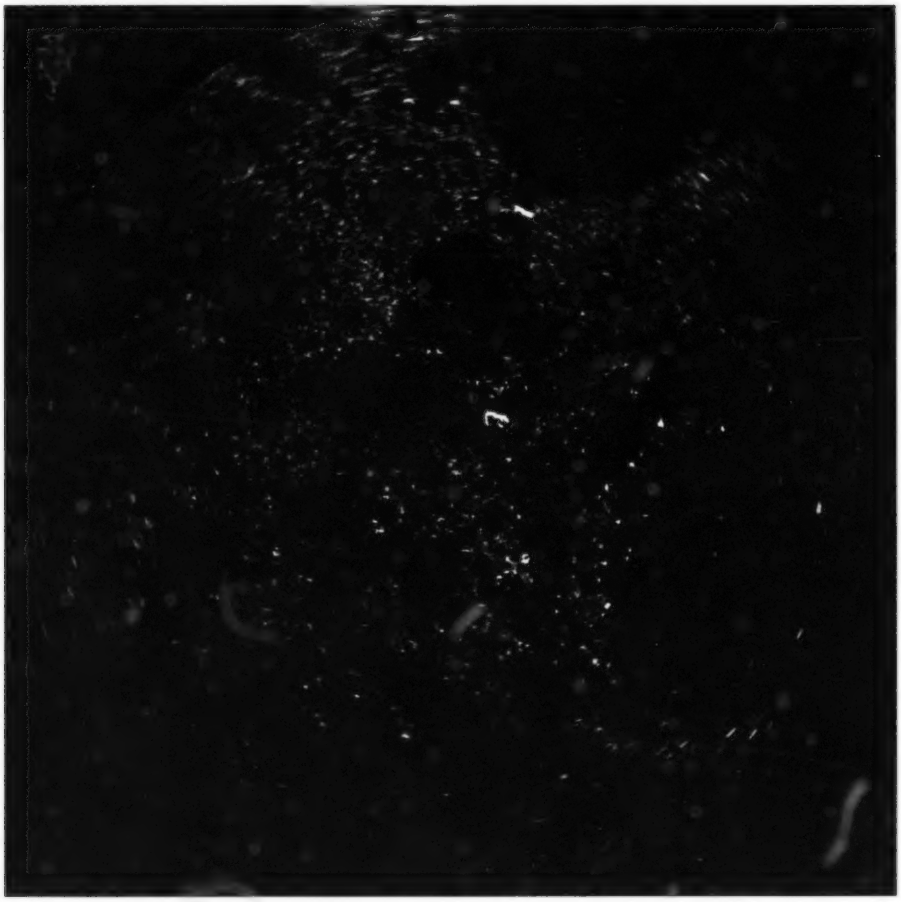


Figure 6.2-2 SRSM image #2

Note: The radar image range is set to 3 nm and OSCFAR is set to ON.

6.3. Inclose (Out of window)

Finally, Figure 6.3-1 presents a snapshot of the view of the horizon as seen from the starboard side of the ship.

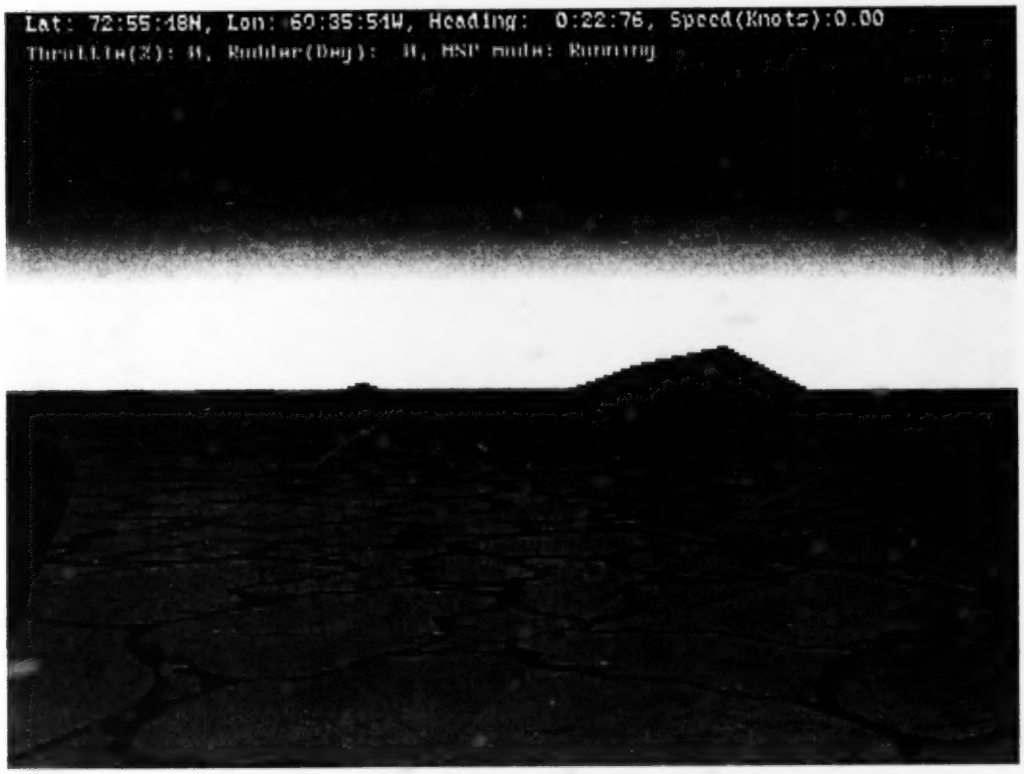


Figure 6.3-1 Inclose "out of window view"

Notice the icebergs in front of the ship. These can also be seen on the simulated radar.

7. Conclusion and Recommendations

Although much progress has been made on the development of the MSP, there are still many things that can be done to improve the system and still some items that are functionally missing in the MSP implementation. These are outlined below.

Main Simulator

- Write Install Program. Develop install CDROM(s) so that all program components would be assured of being installed.
- Automatic Simulator Start-Up. The simulator would start up automatically after MS-Windows is started.
- Add Speech Recognition and Ship Sound to effect a more realistic simulation.
- Add Student Evaluation and Training with prompting of dangerous areas, alarms, warnings, and context-sensitive help.
- Add water depth data.
- Make MSP GUI more intuitive. There are some cases where it is a little awkward.

Scenario Generation Module

- Add image equalize function to SARtoBMP utility. This will make it easier to cut image cookies.
- Incorporate VGM directly into SGM. This will eliminate the possibility of the visual world being out of sync with the analyzed world, i.e. every feature identified in the SGM would be guaranteed to be in the visual world.
- Add 3-D world preview functionality. This would enable the user to "cruise" the world directly without simulating ship motion or without requiring the rest of the simulator.
- Add functions to transpose vector data. This is for the case when the geographic attributes are found to be in error after the imagery analysis has been completed.
- Add utility to import RSAT Data.
- Develop semi-automated SAR analysis to point, click, and detect feature boundaries.

Radar Simulator

- Interpolate polar pixels to synthesize additional resolution when range setting is less than 6 nm. This will remove the pixel replication that occurs on lower range settings.
- Improve sea clutter simulation.
- Simulate cross-polarized radar.

- Modify (SRIC) to retain radar color bitmap adjustment.
- Improve the performance of the Height Map Data Generation in the SGM. The height data is used by the SRSM to create the object shadows on the display. Currently it takes from one to two days to generate this data depending on the size of the SAR data.
- Add ship's track to the SRSM display.

Transient Simulator Module

- Process ice parameters at run time. Currently we only deal with ice type information. Additional parameters such as flexural strength and coefficient of friction, etc. are not dealt with. It will be necessary to implement TSM algorithms.

Visuals Simulator Module

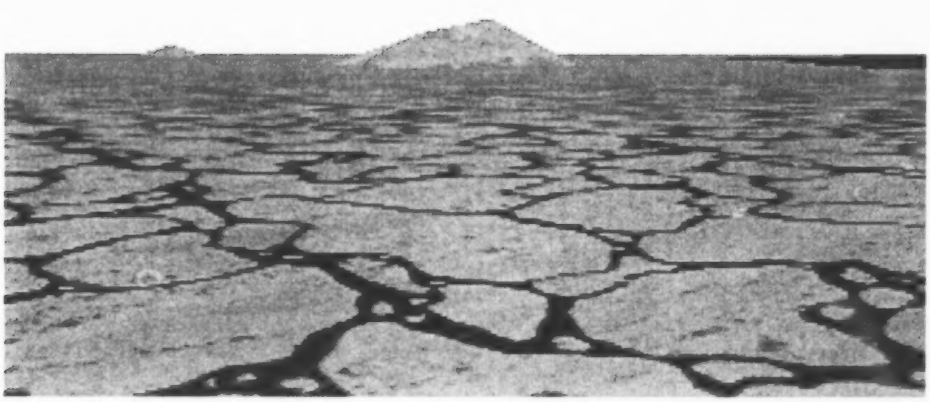
- Multiple out of window views. With the current hardware we can only have a single out of window view. This is because the Tempest Library only supports multiple screens with Primary Image hardware (Piranha Graphics Card). The preferred option is to re-write VSM software in OPENGVS thereby eliminating this problem.
- Add helmet-mounted display (expensive and not necessary at this stage and is mentioned for completeness).
- Add automated visual scenario generation, i.e. script to develop all scenario links automatically.
- Add ship's track to the out of window display.

Appendix A – Additional MSP/VSM Samples

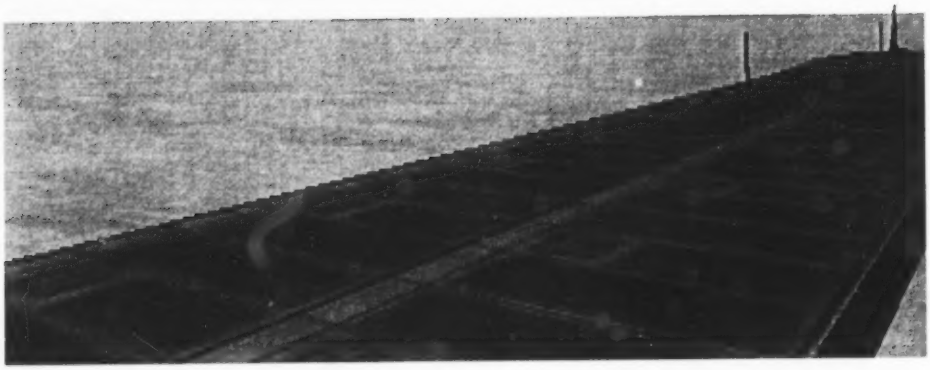
Note: The following are screen captures of a two-screen simulator. By borrowing two Primary Image graphics cards and running them with the VSM it was possible to have two "out of window" displays running concurrently. Additional screen capture software was added to capture the actual screen data. In the pages that follow, the top image represents the left screen and the bottom image represents the right screen.

Lat: 72:55:13N, Lon: 69:35:61W, Heading: 0:00:00, Speed(Knots):0.00
Throttle(Z): 0, Rudder(Dog): 0, MSP mode: Running

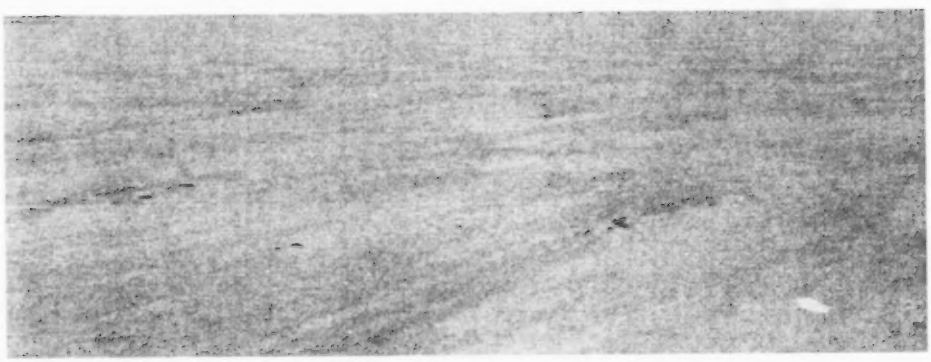
Lat: 72:55:13N, Lon: 69:35:61W, Heading: 0:00:00, Speed(Knots):0.00
Throttle(Z): 0, Rudder(Dog): 0, MSP mode: Running



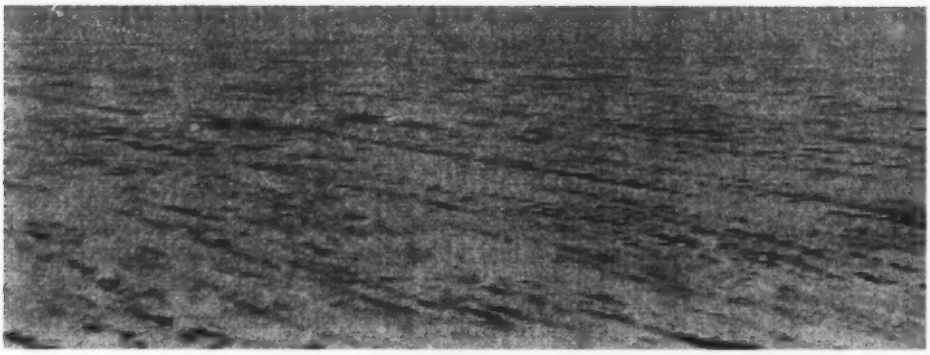
Lat: 73:20:70N, Lon: 78:28:75W, Heading: 0:00:00, Speed(Knots):0.00
Throttle(Z): 0, Rudder(Dag): 0, MSP mode: Running

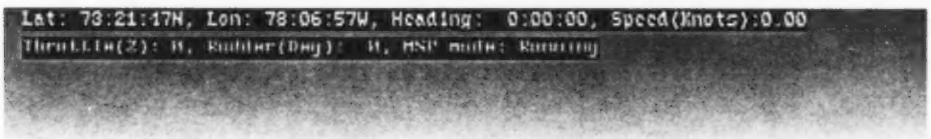


Lat: 73:20:70N, Lon: 78:28:75W, Heading: 0:00:00, Speed(Knots):0.00
Throttle(Z): 0, Rudder(Dag): 0, MSP mode: Running



Lat: 73:21:17N, Lon: 78:06:57W, Heading: 0:00:00, Speed(Knots):0.00
Throttle(Z): 0, Rudder(Deg): 0, MSP mode: Running




Lat: 73:21:17N, Lon: 78:06:57W, Heading: 0:00:00, Speed(Knots):0.00
Throttle(Z): 0, Rudder(Deg): 0, MSP mode: Running